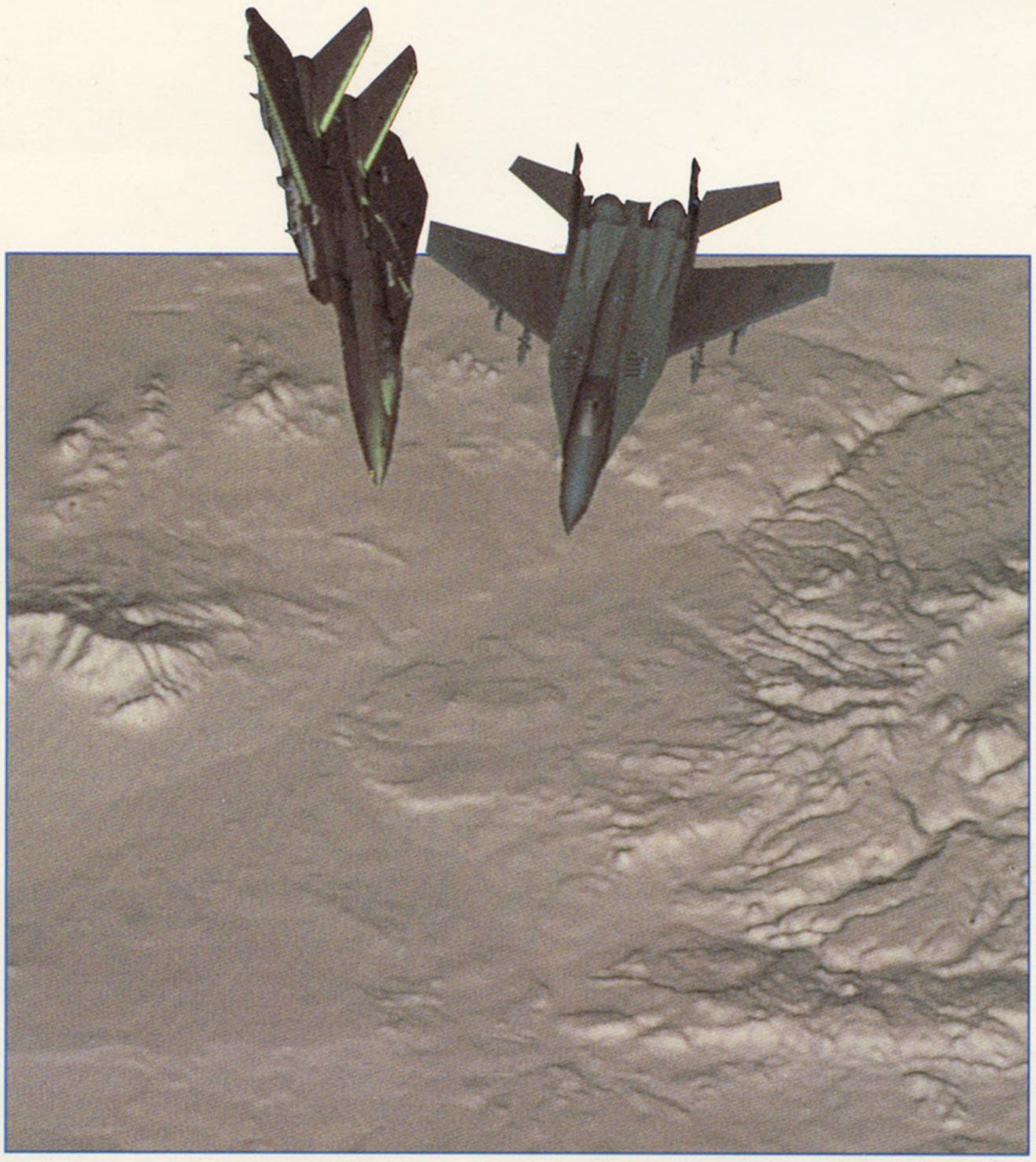
The Magazine of Visual Processing

Issue Number Eleven/Four Dollars

IRIS® universe



Virtual Reality Realized: *PowerVision*™

IRIS: Faster Than a Speeding Cray?

Hidden Charms of Z-Buffer

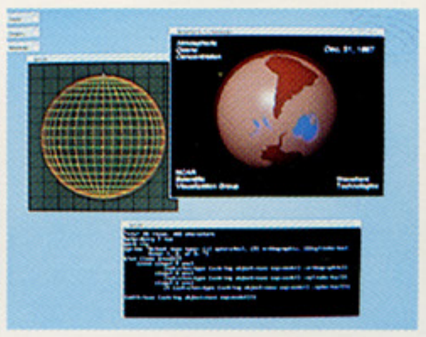
Why did SGI load Personal Visualizer righ

Wavefront's New
t on your copy of 3.2?

Personal reasons!

Directly import and render models built with SDRC's I-DEAS™ solid modeling.

Import model geometry from IGES 3.0, then render it with full reflections and correctly refracted light.



Custom interface modules are fully integrated into the Personal Visualizer for tailor-made imaging presentations.

Expanded libraries increase your imaging flexibility and allows you to make use of thousands of uniquely accurate and tested lighting and surface materials.

WAVEFRONT TECHNOLOGIES

SGI knew that Wavefront was developing a broad range of super friendly visualization options: Rich material and lighting libraries. Powerful 3-D geometry translators for CAD files. And more to come like a Visual "C" application programmer interface and 3-D clip art libraries. All operating under one mouse-driven graphical interface—the Personal Visualizer Core—yours on 3.2 right now. Don't wait to see what the world has been waiting for and *UNIX WORLD* named one of the top 10 products of the year. Call 805-962-8117 for Options orders or information. Wavefront Technologies, 530 E. Montecito St., Santa Barbara, CA 93103.

Unlock Hidden Productivity Tools

Are Your Options Limited Due To Changing Technology?

Because of the rapid pace in hardware development, your software tools are often left by the wayside. You rely everyday on these software applications in your job. When they are locked up because of lack of compatibility between your current hardware and your previous software, productivity is going to suffer. If the application you need isn't available on your workstation, you need TGRAF.

TGRAF Is The Key For Unlocking Software

People who give up their terminals for a stand-alone workstation find themselves either rummaging around trying to beg, borrow, or buy a graphics terminal, or else giving up tools which have helped them do their jobs. Can you afford NOT to have your workstation access all the applications available to you?

TGRAF Does More Than Just Unlock Your Software

TGRAF software lets you connect your workstation to existing applications running on other hosts. It does this by emulating Tektronix 4107 and 4125 terminals, the most widely used graphics protocol today. Both serial (RS- 232 or modem) and network host connections are supported, as are task-to-task connections for Tektronix-compatible software running locally on your workstation.

Call Us Today For A Free TGRAF Demonstration

Don't keep your software tools locked away. Call us to set up a demonstration. See for yourself what unlocking your software can do.
USA 1-800-426-2230, (in California and outside the U.S. 408-446-1919), Fax 408-446-0666, or write to: Grafpoint, Workstation Products, 1485 Saratoga Ave, San Jose, California, 95129.

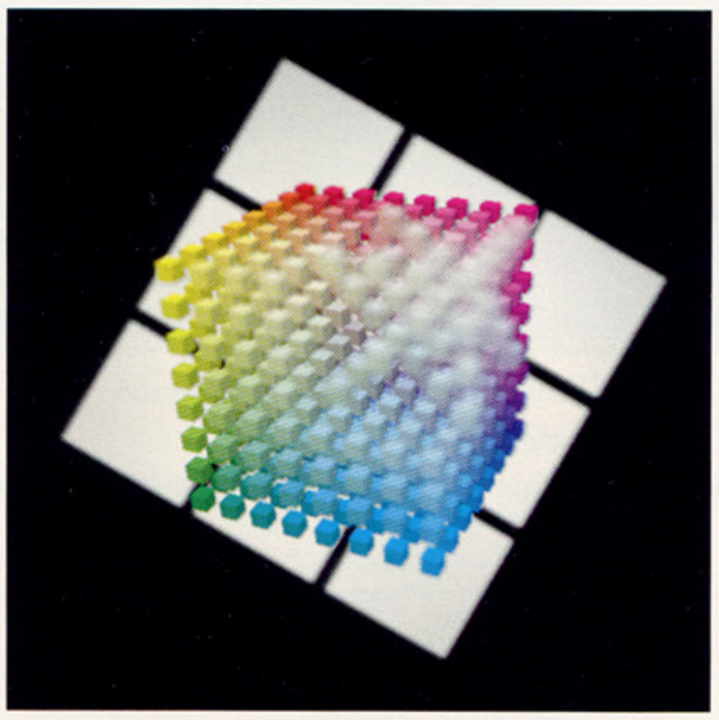
GRAFPOINT™

Grafpoint manufactures a comprehensive line of powerful Tektronix terminals for PCs, PS/2s, workstations, and Macintosh computers.

IRIS

universe

6



8

17

FEATURES

8 IMITATING LIFE
By Joshua Mogal

PowerVision, a new graphics architecture, offers users polygon anti-aliasing, texture mapping, and faster vector anti-aliasing than ever before. Add polygon rates rapid enough to enable Out The Window simulation, and you see why the product is considered a major evolutionary step for the Geometry Pipeline.

17 THE RIGHT TOOLS FOR THE TASK
By Laurence Feldman

Employing a network of computers including a Macintosh, a 386 based machine, an IRIS-4D, a Sun-4, a Cray-2, and a Cray-YMP, the author assesses the role each can play in optimizing visual simulations.

28 DATABASE VISUALIZATION
By Chuck Molyneaux

The new *Visual Database Lab* makes it possible for application developers to blend database information with images and sound in an interactive, multi-windowed, multi-processing, display environment.

31 THE HIDDEN CHARMS OF Z-BUFFER
By Kurt Akeley

Little known tricks for using IRIS Z-buffer hardware to handle complex operations such as applying decals, highlighting surface tesselations, and generating line drawings with hidden lines removed.

DEPARTMENTS

6 SEQUENCE
By Rolf van Widenfelt

A *geometry movie* showing the shape optimization of an automobile connecting rod.

38 SOLUTIONS
By Pramod Rustagi

A technique for simulating wireframe degeneration without having to create alternate data structures

41 COMMUNITY FORUM
By Gaye Graves

New software improves the client's ability to participate in the design process.

42 PRODUCT BRIEFING
By Gaye Graves

Software tools for the IRIS transform RGB images into color separations used in four color printing.

What's missing in your FORTRAN problem-solving environment...

...IMSL's Mathematical and Statistical Libraries

If you rely on FORTRAN as a problem-solving tool, you recognize the value of the small library of intrinsic functions that came with your FORTRAN compiler. You wouldn't think of coding a routine for a square root, sine, or cosine; you simply call it from the intrinsic function library. But what about the many other standard, more complex mathematical and statistical capabilities you so often need?

What's missing from FORTRAN is a complete, comprehensive library of mathematical and statistical routines...The IMSL Libraries.

IMSL's MATH/LIBRARY, SFUN/LIBRARY, and STAT/LIBRARY comprise more than 800 FORTRAN subroutines covering a broad range of mathematics and statistics. Linear and non-linear systems, differential equations, eigensystems, regression, correlation, and special functions are just a few of the capabilities that can be literally at your fingertips with IMSL's FORTRAN subroutine Libraries.

And that's just part of the story...combined with online documentation and the high-level FORTRAN-like procedures of MATH/PROTRAN and STAT/PROTRAN, IMSL provides the comprehensive mathematical and statistical tools that are missing in your FORTRAN program development.

When you call an IMSL subroutine, you're calling on expert design and development, rigorous testing, and proven reliability demonstrated by hundreds of thousands of hours of use by IMSL customers around the world. To IMSL users this means shorter development time, more robust solutions, and lower costs.

For almost 20 years, IMSL customers in industry, government, research, and education have chosen our high-quality, high-value FORTRAN Libraries. We'd like to talk to you about how the IMSL Libraries can enhance your FORTRAN application development environment. We have what's missing, contact us today: toll-free **1-800-222-IMSL**, or in Texas (713) 782-6060; 2500 Permian Tower, 2500 CityWest Blvd., Houston, TX 77042-3020; FAX: (713) 782-6069.

My interest is in software for:

☐ Mathematics ☐ Statistics
☐ Special Mathematical Functions ☐ Online Documentation

Name Title

Organization

Department

Address

City/State/Country Postal Code

(Area Code) Phone Telex or Fax

Computer Type Operating System

My need for ordering IMSL software is:

____ Immediate ____ 3 Months ____ 6 Months ____ 1 Year

IMSL, Inc.
Marketing Communications
2500 Permian Tower
2500 CityWest Boulevard
Houston, Texas 77042-3020
1-800-222-IMSL
SG9004

IMSL®
We make FORTRAN more useful

IRIS *universe*

Editor
Douglas Cruickshank

Contributing Editor
Gaye Graves

Page Design
Julia Wieczkowski

Digital Images
Monica Schulze and Paul Haeberli

Color Separations
Star Graphics

Typesetting
Lauren Langford Typography

Printing
Hatcher Trade Press

Magazine Design
Nancy Jorgenson

Editorial
Silicon Graphics, Inc.
2011 N. Shoreline Blvd.
Mountain View, California 94039
415/335-1727
<*dougc@sgi.com*>

Circulation
Monica Schulze
Silicon Graphics, Inc.
2011 N. Shoreline Blvd.
Mail Stop 415
Mountain View, California 94039
<*monica@sgi.com*>

Advertising
Ardith Lowell
MKT.
2755 Campus Drive #247
San Mateo, California 94403
415/341-9681

Publisher
Mark Compton

ON THE COVER

A sixty square mile quadrangle of the Albuquerque, New Mexico region with an F-14 and Mig-29 in "Top Gun" scenario. REP_TILE produced the image on a Cray-2 in approximately 1 to 3 seconds per frame (Courtesy of the Air Force Supercomputing Center, Kirtland Airforce Base, New Mexico.)

About This Issue

The Magnificent Adventure

"Here it comes into our lives.... It enters because we want it, we need it, a human machine in a way no other has been.... To remain ignorant to it is to step aside from our own century's contribution to all the other magnificent intellectual adventures that have exalted the human spirit, beginning, perhaps, with the invention of language itself.... In the distance, these troubled times may be remembered best for the invention — the inevitable invention — of an instrument to give us heart, for it liberates and magnifies the human property that has always served us best, our own intelligence."

— Pamela McCorduck
The Universal Machine: Confessions of a Technological Optimist

It is often said that humans treat the automobile as an extension of themselves. What we've seen in the last few decades is the introduction of a much different means of personal extension, the "human machine" of which Ms. McCorduck writes so eloquently. While the automobile gave us physical mobility and diminished geographical distances, the computer has provided something even more enriching — intellectual and creative mobility and the shrinking of *conceptual* distances.

The challenge faced by the developers of modern technology is that of constantly expanding the capabilities of machines, so that technology keeps pace with the minds that use it. The introduction of the new graphics architecture, *PowerVision*, makes an important contribution to that goal, as Joshua Mogal explains in this issues' lead article, "Imitating Life."

While *PowerVision* is an evolutionary step for the Geometry Pipeline, Kurt Akeley's tips for fully exploiting the Z-buffer hardware serve to magnify the already considerable abilities of the IRIS. Likewise, Laurence Feldman's keen assessment of "The Right Tools for the Task" enlarges our knowledge of how best to apply hardware and software by comparing the strengths (and weaknesses) of different computers in optimizing 3D simulations.

Also in this issue, systems engineer Chuck Molyneaux discusses the new *Visual Database Lab*, a display product which simplifies a complex Visual Processing requirement, the need to capture video signals and mix live color images.

On page six, we introduce a new feature which will appear in every issue. It's called "Sequence." In the first installment, Rolf van Widenfelt "narrates" a *geometry movie* as he guides us through a shape optimization simulation.

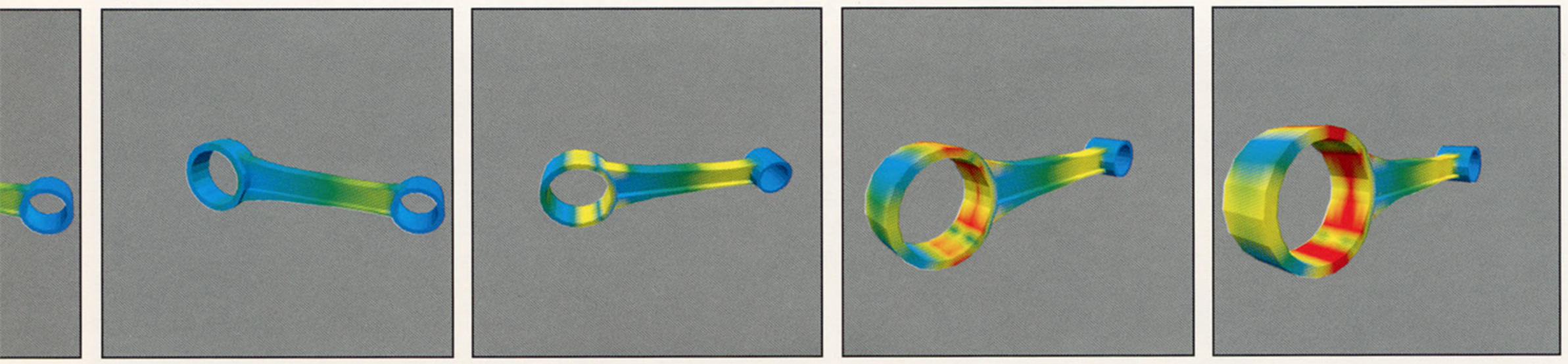
Intellectual liberation and the magnification of intelligence; these were the goals of Visual Processing at its inception and continue to be the goals which propel its development. As this issue reveals, technical limitations are rapidly disappearing. The full realization of our ideas and creativity comes closer via the extraordinary tool that is Visual Processing.

— *Douglas Cruickshank, Editor*

IRIS Universe: The Magazine of Visual Processing, is published quarterly by Silicon Graphics, Inc., and is dedicated exclusively to the needs and interests of the Visual Processing community. Please send address changes to "IRIS Universe, Silicon Graphics, 2011 North Shoreline Boulevard, Mail Stop 415, Mountain View, CA. 94039-7311". Subscriptions are available upon request to qualified users. Others may subscribe at the following rates: $14 per year (USA), $16US per year (Canada), $26US per year (Overseas). Fill out the postage-paid subscription card in this issue or call (415) 962-3320. Correspondence regarding editorial (press releases and product announcements) should be sent to "Editor, IRIS Universe, 2011 North Shoreline Boulevard, Mail Stop 415, Mountain View, CA 94039-7311". Phone: (415) 335-1727. E-mail: *dougc@sgi.com*. Letters to the *IRIS Universe* or its editors become the property of the magazine and are assumed to be intended for publication, and may be so used. Correspondence should include the writer's full name, address, and telephone number. All letters may be edited for clarity and/or space.

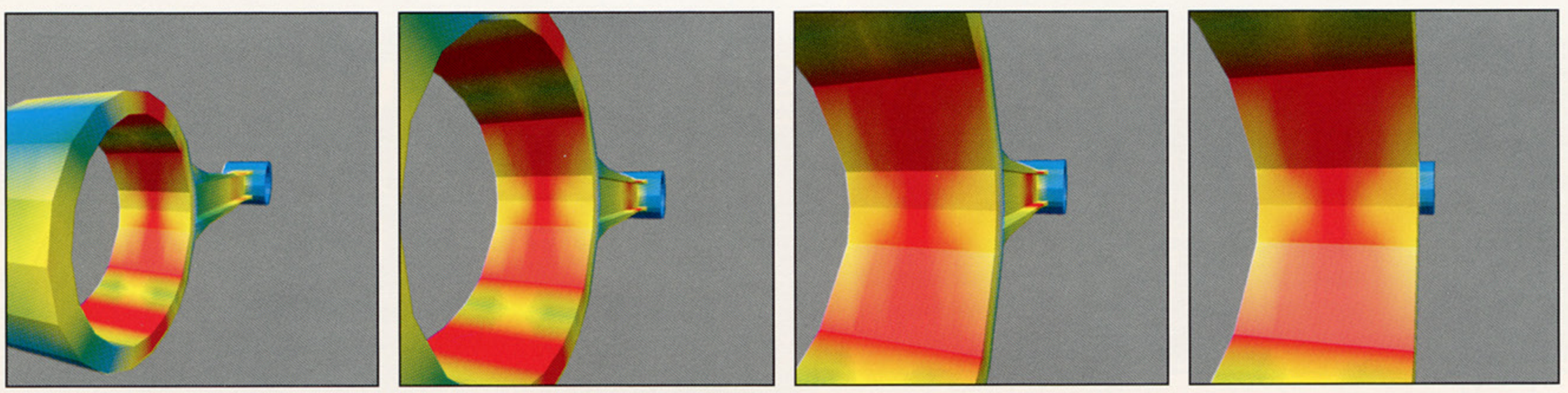
SEQUENCE

By Rolf van Widenfelt



This sequence shows the progress of a finite element shape optimization simulation for an automobile connecting rod. The simulation's objective is to create an optimal part based on a given initial design and a set of geometric and stress constraints. Our objective is to display the progress of this simulation. Two programming techniques enhance the display's effectiveness.

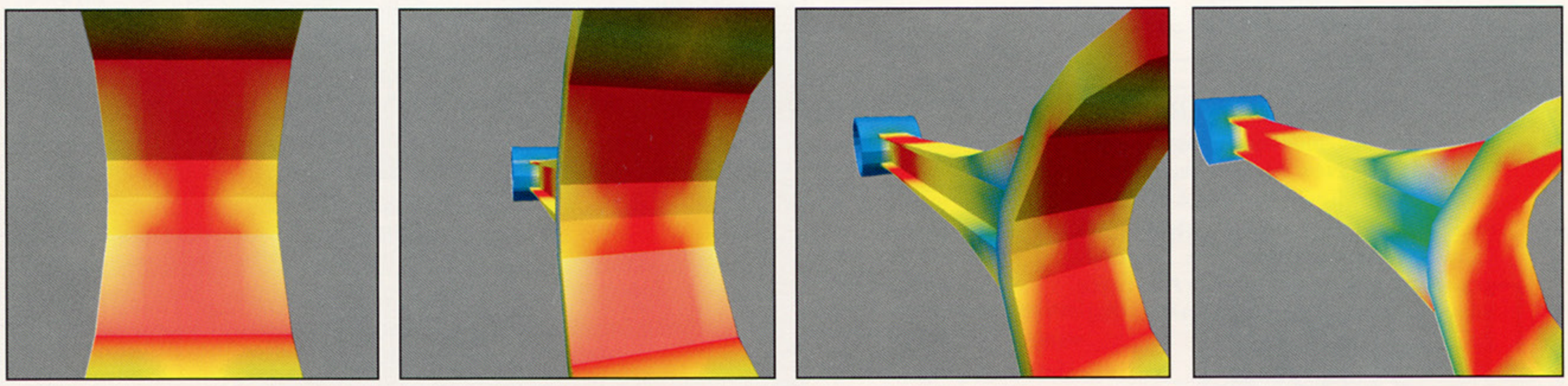
We want to indicate stress on the rod with surface color, but coloring the surface according to stress alone makes it difficult to discern the rod's shape (see last frame of sequence). Shape is more effectively indicated using the Graphic's Library's (GL) lighting facility. The *lmcolor()* primitive makes this possible. Instead of directly specifying the color, *lmcolor()* allows us to specify a *material color* at each vertex. The *material color* is a surface property utilized by GL lighting model, together with light specifications and viewer position, to calculate the final color that is displayed. In this simulation blue material indicates low stress, red indicates high stress. Intuitively, one can think of a plastic connecting rod under illumination where the color of the plastic represents the stress.



The program stores a number of frames, each representing one step of the simulation. A frame is stored not as a pixel image, but as 3D geometry. This allows us to playback frames in sequence, as with a pixel movie, but with the added capability of changing position, orientation, and view. The ability to sequence through a set of geometry frames while maintaining interactive control of the view can be a powerful method for visualizing time-varying or parameterized data sets. We call this a *geometry movie*.

The connecting rod contains 828 polygons, and is displayed at 15 frames per second on an IRIS 4D/80GT. The lighting uses two infinite lights with an infinite viewer. These 1000 X 1000 pixel images were taken directly from an IRIS screen. (Connecting rod data courtesy of Michael Long, Cray Research, Inc.) ■

Rolf van Widenfelt is a member of the technical staff, Advanced Systems Division, Silicon Graphics.

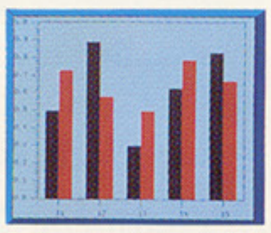


TRUE GENIUS IS THE SIMPLE EXPRESSION OF COMPLEX IDEAS.

You spend months, maybe years, working out every possible intricacy for a new software application. And somehow you're supposed to create a graphical user interface that makes it easy for somebody to understand it all with just a few clicks of a mouse, a few objects on a screen.

Perhaps the real genius in application development isn't the complexity of the task, but the simplicity of its use.

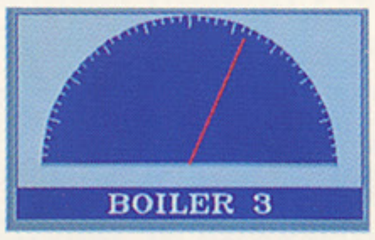
DataViews® was created with the belief that the most powerful applications are those that are most easily understood. Which is why we offer the most complete graphical user interface development environment on the market today.



Simply stated, DataViews can provide you with an exceptional set of tools for making complex applications easily understood.

Tools that enable you to create dynamic displays of real-time data. Tools that offer you total portability between major operating platforms. Tools that are relevant to almost all application environments including telecommunications, finance, manufacturing, and defense.

But most importantly, our tools can save you endless hours creating and revising code for the visual interface between your application and the end-user.



And by spending less time on your graphical interfaces, you'll have more time to create your application.

Which, after all, is where your true genius shines.

DataViews
V.I. Corporation

V.I. Corporation, Amherst Research Park, Amherst, MA 01002

For more information about DataViews call 413-253-4270.

SiliconGraphics
Computer Systems

feature

IMITATING LIFE

The introduction of PowerVision, *a major new graphics architecture, raises the standards for real-time rendered images*

BY JOSHUA MOGAL

Whether or not art imitates, interprets, or distills life, Computer Graphics has, since its genesis, been used as a tool to model reality. The technique of modeling is itself an art; the art of discovering why the world is as it is and of attempting to recreate and understand form and behavior. Until the advent of computer graphics, modeling was restricted to such artistic endeavors as painting and sculpture and to scientific studies such as those by Leonardo DaVinci of the techniques of flight (Daedalus Image), mechanics, and biology. Today, however, we need not take months or years to create such models thanks to the wealth of modeling tools available through computer graphics.

Silicon Graphics has, as a company, focused on providing graphics computers that render models as realistically as possible while retaining interactivity. Interactivity assures that the tool is usable by all people, not simply those with an infinite supply of patience. Most recently, in July of 1987, Silicon Graphics announced its second generation graphics architecture, the GT. The GT added smooth Gouraud shading, more realistic lighting models, line anti-aliasing, and alpha blending for transparency, while pushing the performance limits to offer rendered throughput of up to 100,000 independent 4-sided, lighted, clipped, and z-buffered polygons per second (or even 135,000 polygons/second if triangular meshes were used).

These improvements in rendering quality and throughput enabled tremendous growth for graphics workstations in the fields of Computational Chemistry, Visual Simulation, CAD/CAM, Animation, and Scientific Visualization. Previously, users had been limited to viewing graphic models in wireframe or flat shaded mode to preserve interactivity, reserving high quality shading for static images. With the advent of the GT, they could view models with smooth shaded surfaces and transparency, improving their ability to visualize data and more effectively understand their models. Even at this level of performance, though, the GT hardware could not match the "state of the art" rendering software (which isn't necessarily concerned with interactivity).

Introducing *PowerVision*™

Thus, *PowerVision* was introduced, a family of graphics supercomputers based upon the third major graphics architecture from Silicon Graphics. IRIS *PowerVision* systems incorporate the IRIS *VISION*™ graphics subsystem, based upon a new architecture combining both Single Instruction, Multiple Data (SIMD), and Multiple Instruction, Multiple Data (MIMD) parallel processing techniques. Up to eighty-five custom, programmable VLSI ASICs containing 4.7 million gates have been newly designed and incorporated, utilizing 1.0 micron CMOS technology. Enabling a plethora of new features and significant performance improvements, this new architecture is a revolutionary design based upon the fundamentals of the Geometry Pipeline first implemented by Jim Clark at Stanford University in 1979[1].

The new system is designed to extend the limits of graphics performance and functionality, yet to maintain compatibility with and an entry price point near that of systems based upon the

GTX graphics subsystem. Research was conducted in 1988 in each of the various market segments making up the high end of Silicon Graphics' graphics supercomputer business. The research findings indicated broad demand for a variety of features and performance improvements. Among the requirements most often specified were texture mapping, improved fill rates, polygon anti-aliasing, faster point and vector anti-aliasing, polygon rates fast enough to enable Out The Window visual simulation, and faster screen-clearing rates.

System Architecture

The *PowerVision* architecture enables this feature set, meets the performance requirements, and more. In the first major effort to extend the capabilities of Silicon Graphics' Geometry Engine subsystem, *PowerVision* has been microcoded to perform both Geometric *and* Image Processing operations right in the pipeline. This frees the CPU subsystem for analysis while image or volume operations, previously possible only in the CPU, are performed in the Graphics subsystem. System performance has been boosted to enable the display of up to 1,000,000 polygons and 1,000,000 anti-aliased vectors per second, with the screen clear time reduced tenfold to 0.78 milliseconds.

One of the major benefits of the new SIMD parallel architecture (fig.1) at the front end of the pipeline is that the system can be programmed to perform general purpose floating point processing on multiple, simultaneous data values. Using the new *IMAGEVISION LIBRARY* provided with the system, new opportunities are created for image and volume processing applications, providing hardware support for functions that had previously been possible only in software. In certain cases, *IMAGEVISION* operations can be performed at up to 128 MFLOPS.

The *IMAGEVISION LIBRARY* is a set of routines callable from either C or FORTRAN and runs on the same processors as the *Graphics Library*™ (GL). To execute *IMAGEVISION* operations, the four parallel Geometry Engines are programmed with a set of operations different from those used by the GL. Each of these Geometry Engines has 8K Words (32 bit) of scratchpad memory. This memory may be partitioned into arrays of elements of arbitrary length (up to 1024) and width (up to 4, although multiple passes allow more than 4). These elements are treated like vectors.

When performing element operations, each operand or destination element (i.e., V3 = V1 + V2) may reside either in user memory (on the CPU side) or in the scratchpad memory immediately associated with the processors. To maintain optimum performance, all of the vectors in each operation should be resident in the scratchpad. Some of the major operations included in the *IMAGEVISION LIBRARY* are shown in Table 1.

Other Image and Volume operations such as the use of convolving filters may be stored in the texture memory and applied by the Image Memory Processors (IMPs - shown in Figure 1 as PPs) in the Raster Subsystem section of the *VISION* graphics subsystem, while more generalized applications may still be run in the parallel POWERpath CPU subsystem.

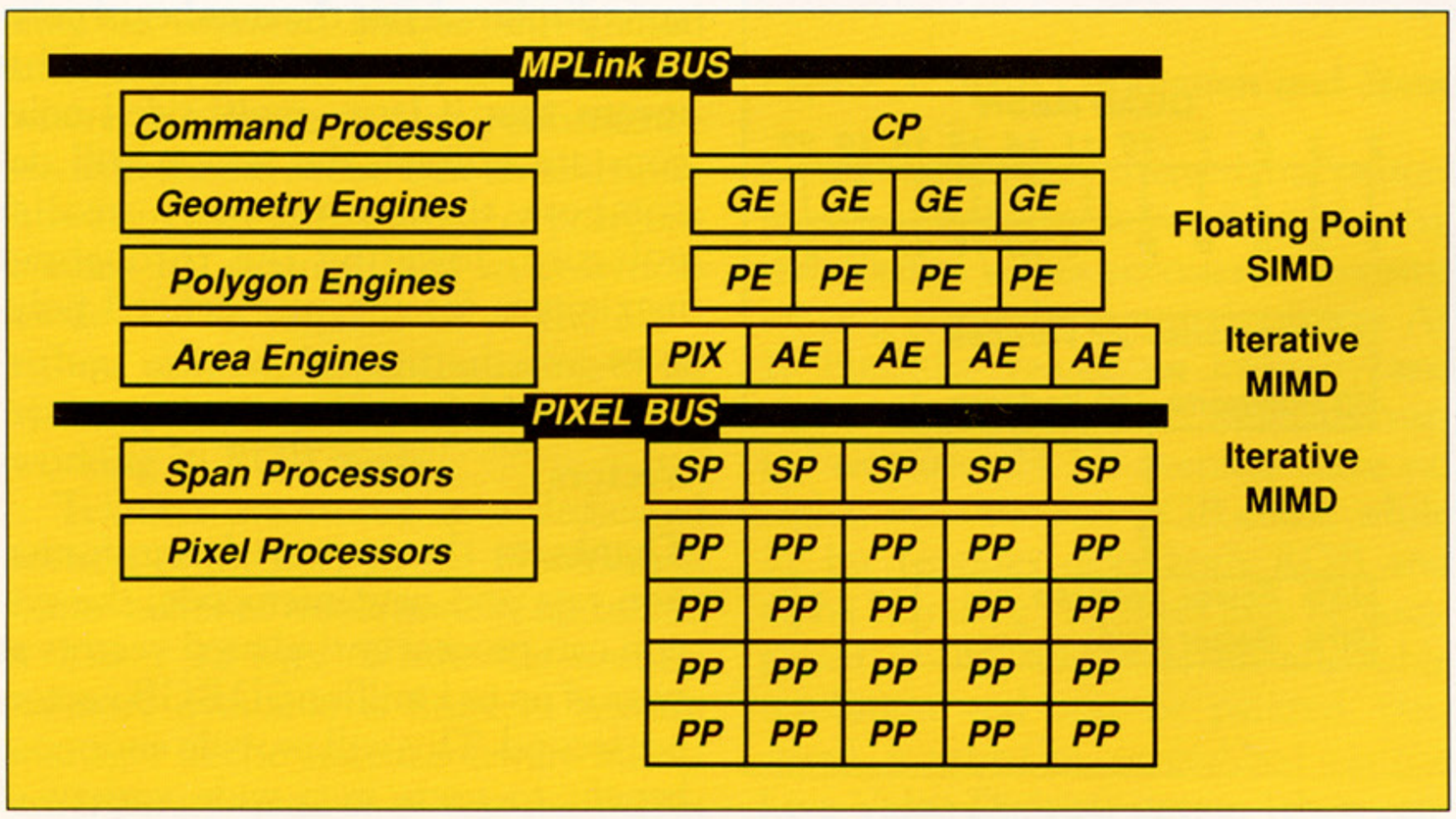


Figure 1: Graphics Subsystem Architecture (simplified)

Geometric Performance Maximization

Polygons

For geometric operations, it may be inferred from the architectural diagram in Figure 1 that due to the number of processors in the top of the pipeline, the system works ideally on polygons of either three or four vertices. Indeed, the architecture has been optimized for these types of polygons and thus can process them at up to an order of mag-

iladd	add two vectors	*ilifft*	inverse FFT
ilsub	subtract two vectors	*ilsqr*	square of elements of a vector
ilmul	multiply two vectors	*ilsqrt*	square root of elements of a vector
ilvsmul	multiply a vector by a scalar	*ilsfft*	forward FFT
ilmixmul	multiply a real and a complex vector	*ilmov*	move vector from one vreg to another
ilabs	absolute value of elements of a vector	*ilset*	set the elements of a vector
ilsin	sine of elements of a vector	*ilramp*	set elements of a vector to an arithmetic progression
ilcos	cosine of elements of a vector	*ilgramp*	set elements of a vector to a geometric progression
ilatn2	arctangent of elements of two vectors	*ilrotate*	rotate a vector around an element
ilatan	arctangent of elements of a complex vector	*ilsaxpy*	multply a vector by a scalar and add to a vector
ilexp	exponential of elements of a vector	*ilclip*	clip the elements of a vector against a value
illog	logarithm of elements of a vector		

Table 1: ImageVision Library Functions

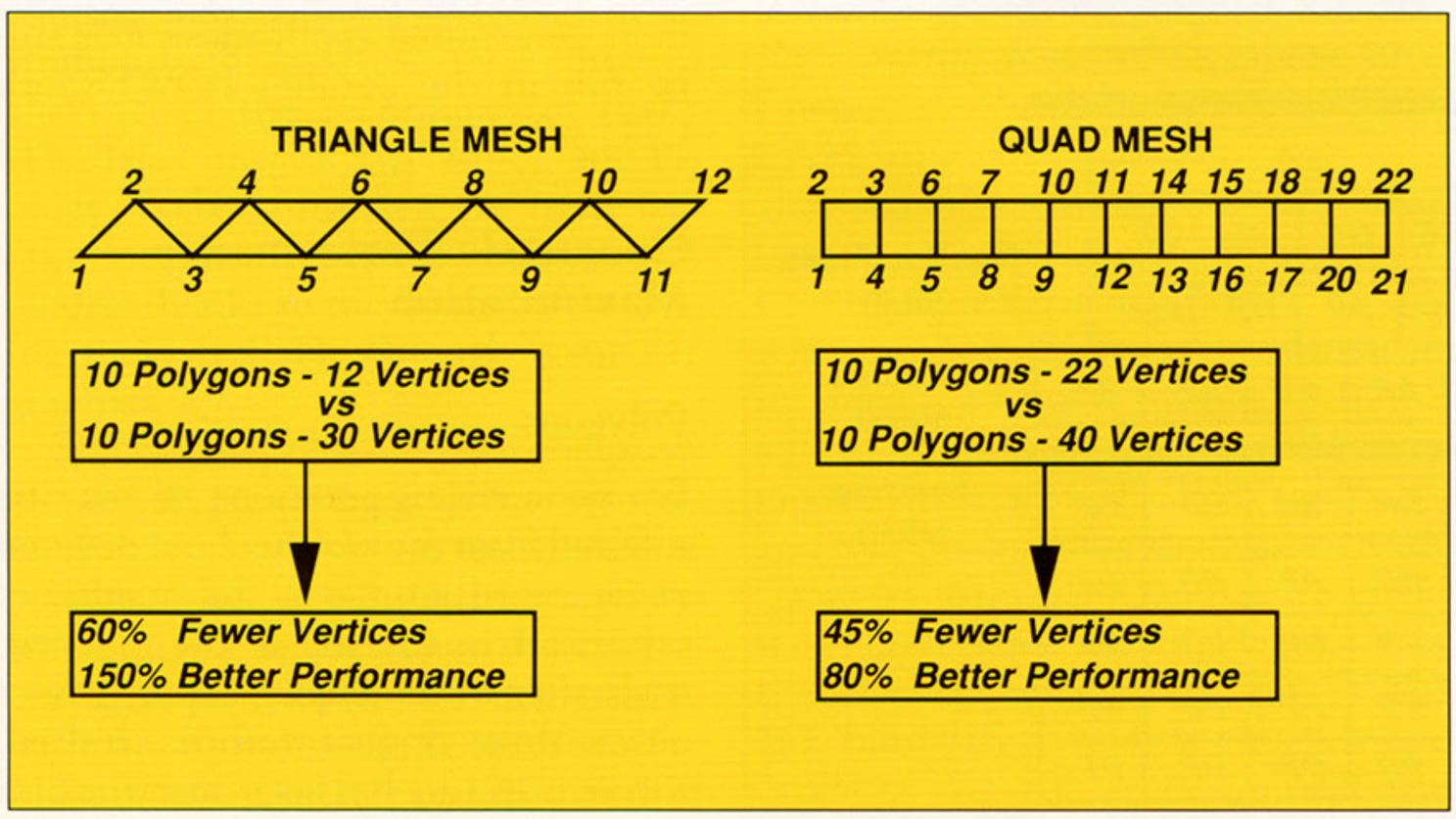


Figure 2: Triangle Meshes and Quad Strips

nitude faster than previous architectures. Polygons with more than four vertices would require the pipe to process the first four, and then wait for the rest before completing its operations.

Thus, to achieve maximum throughput for the *PowerVision* subsystem, polygons should, whenever possible, be arranged in strips of triangles or quadrangles (see Figure 2). Using the Triangle Mesh and Quad Strip primitives, the top performance figures of 1 million triangles and 400 thousand quadrangles per second are achievable.

As indicated in the diagram, triangle meshes and quad strips can improve triangle and quadrangle throughput by 80-150% over the rate possible with independent polygons simply by reducing the number of vertices requiring transformation by up to 60%. *This is not the actual performance speedup*. In actuality, the performance gain is much better, as the SIMD architecture processes independent triangles one at a time and meshed triangles *four* at a time. This results in an actual meshed triangle performance of 1 million polygons per second.

Note that all polygon throughput numbers represent the number of polygons displayed, not the number of polygons supplied to the system. This means that if large, multi-sided polygons are utilized, the system will decompose them into many smaller polygons, lowering the throughput maximums for the user defined polygons.

Vectors

Thanks to the improved transformation rate and new microcode, the system can process anti-aliased vectors at rates of up to 1 million 3D RGB vectors per second. This will provide enormous benefit to users in a wide variety of areas, but particularly in Computer

Aided Molecular Design (CAMD). CAMD scientists display stick and ball molecular models with hundreds to thousands of atoms and thus many thousands of 3D vectors. To maintain interactivity, 200K-300K vectors would be required. *PowerVision* has gone beyond the minimum requirements to adequately cover the needs of this market into the next generation of software.

Even with these performance improvements, certain users will still be concerned about the *quality* of *PowerVision*'s anti-aliased vectors. Anticipating this, *PowerVision* was designed to allow programmability of the anti-aliasing filter funtions. These filters are stored in a tabular format in the system and are programmable by *Silicon Graphics, not by users*.

Texture Mapping and Anti-Aliasing

PowerVision is making its mark as the first standard workstation to offer *real-time* natural looking texture mapping and anti-aliasing of *all* graphics primitives, from polygons to vectors to points. Using *PowerVision*, primitives may be textured and/or anti-aliased selectively to enable the user to decide where the image quality/performance tradeoff should occur. One should note, however, that anti-aliasing of polygons cannot be accomplished in colormap mode due to restrictions of the algorithms utilized, but anti-aliasing of all system primitives is available for those working in RGB mode.

Textures are stored in a dedicated section of memory in the *PowerVision* frame buffer at resolutions of up to 256 x 256 x 32 bits. While the texture memory available is finite, a texture manager in the system software allows an arbitrary number of textures across processes. At reduced texture depths, no texture memory is saved, but performance is enhanced. For 8 bit depths, the texture values represent intensity, varying across the textured polygon. At 16 bits, the values are intensity and alpha allowing blended intensity varied textures. 24 bit textures have no alpha, but offer full RGB color, while 32 bit textures offer both RGB and alpha blending. Textures may be comprised of any image at an allowable resolution and may be painted, rendered, scanned in, or grabbed via the Live Video Digitizer option (then captured from the screen).

t2	assign texture coordinates to a vertex
tevbind	select a texture environment function
tevdef	define a texture environment function
texbind	select a texture function
texcontour	automatically generate texture coordinates
texdef	define a texture function
texscale	change mapping of a texture to geometry

Table 2: GL calls for Texture Mapping

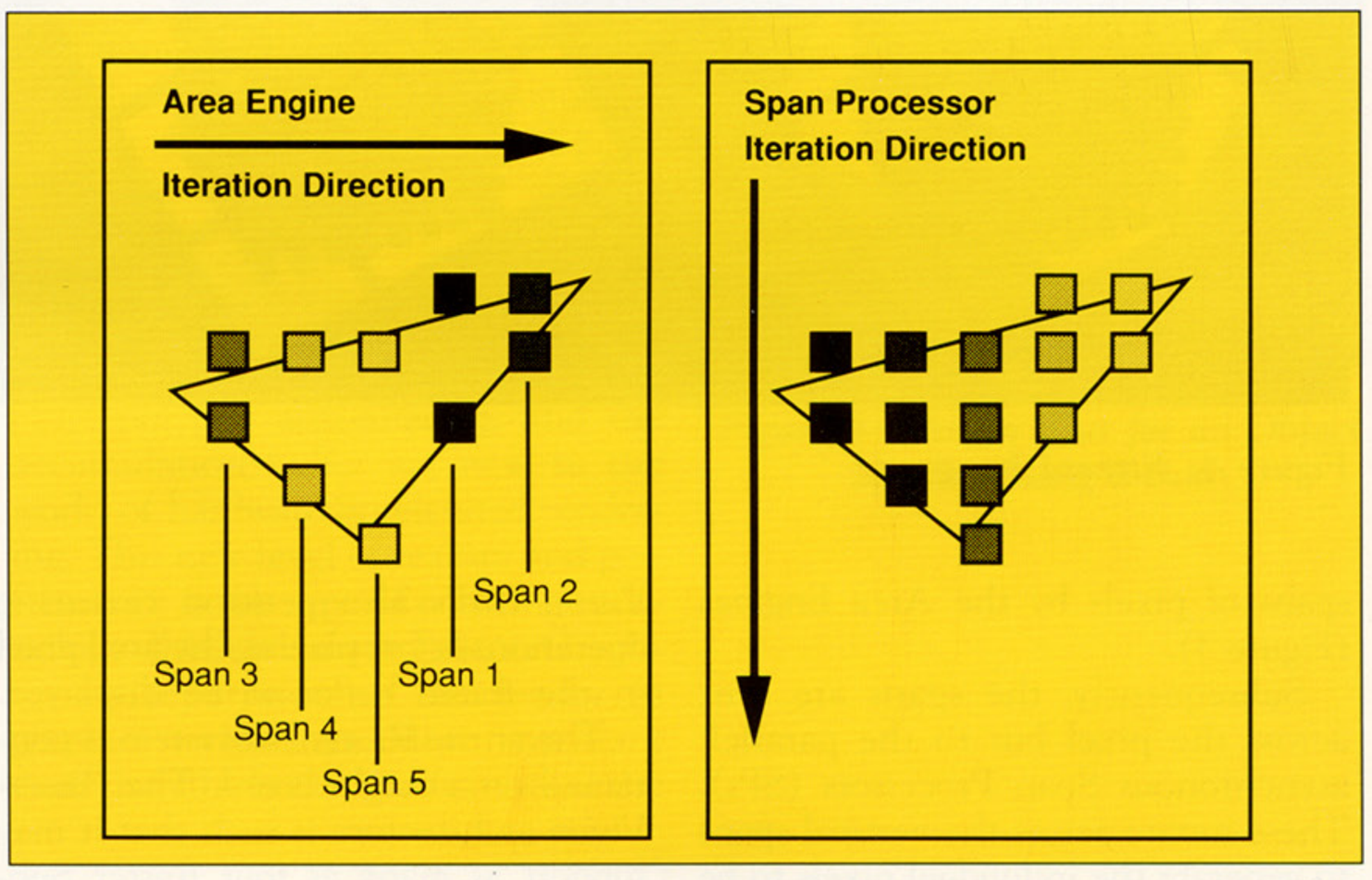
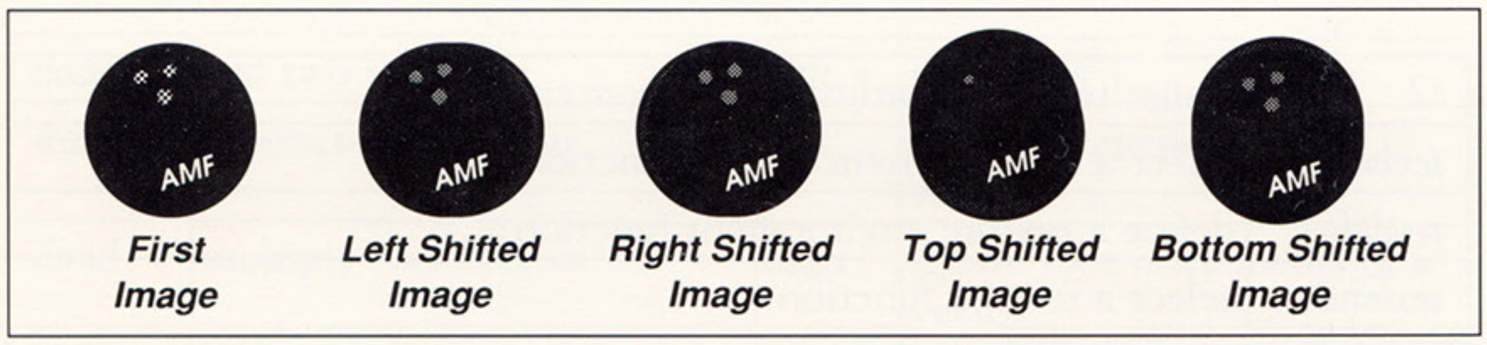


Figure 3: Span Conversion and Span Iteration

The POWER Frame Buffer Expansion option doubles the frame buffer memory, the texture memory, and the pixel fill rate. All in all, there are eight new GL commands for dealing with texturing as shown in Table 2.

The actual texture mapping is performed at the tail end of the pipeline in the Image Memory Processors (IMPs). The IMPs are located in the *VISION* graphics raster subsystem. As polygons progress through the pipeline, they are lighted, transformed, and clipped in the geometry subsystem. Just before leaving, they are turned into vertical



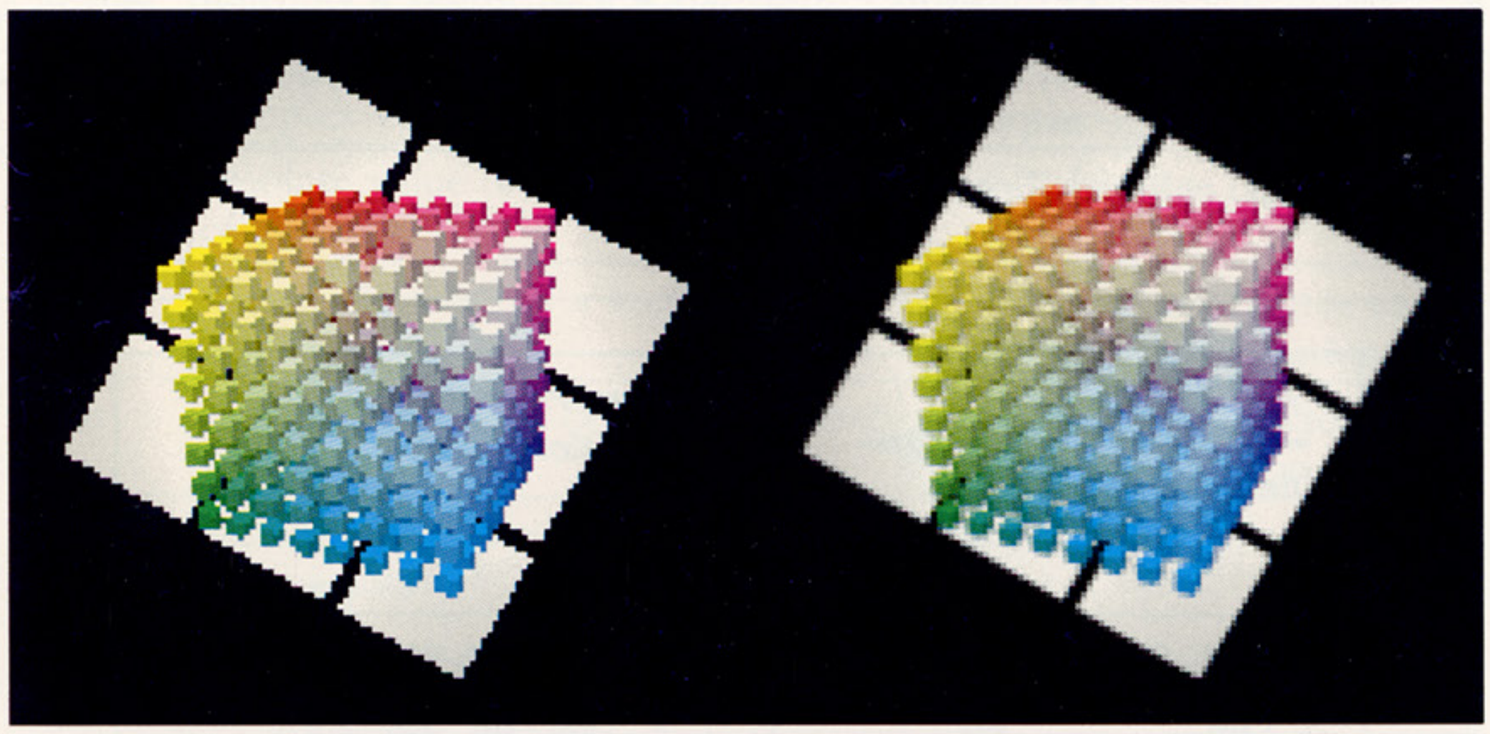


Figure 4: Jittered Images

spans of pixels by the Area Engines (Figure 3).

Subsequently, the spans are sent across the pixel bus to the parallel, asynchronous Span Processors (SPs). These iterate across the vertical spans to generate the individual pixels to be displayed for each polygon. Each SP will assign colors and Z values to each of the pixels in its span, then hand each pixel off to one of the four IMPs associated with that SP. The IMPs are each in control of certain pixels in the frame buffer. The actual pixels in the frame buffer are interleaved so that for every five pixels across a given scan line, each of the five pixels is managed by a different SP. This is comparable to the organization of the frame buffer in the GT. The IMPs in the *PowerVision* subsystem, like those of the GT, are responsible for performing the Z-Buffer tests and Alpha blending on the pixels. Unlike the GT, however, the *PowerVision* IMPs also perform texturing operations *before* placing the final pixel in the frame buffer to be displayed.

The entire Raster Subsystem is contained on a single board. The *PowerVision* architecture is such that it may support as many as four Raster Subsystems permitting a range of systems with from 5 to 20 spans. Since the SPs are the limiting factors in pushing pixels to the screen, it follows that the more Span Processors, the faster the pixel fill rate. Thus, by adding spans (or RM2s), a *PowerVision* system may fill from 100 million pixels per second (with 5 spans) to 400 million pixels per second (with 20 spans). For the initial release, 20 span systems will not be available.

Additional Raster Subsystems not only add SPs, IMPs, texture memory, and frame buffer memory, but increase the number of available overlay and underlay planes, and enable accelerated performance for the *FlexScene* accumulation buffer.

FlexScene

The *FlexScene* accumulation buffer is a feature which adds tremendous benefit to *PowerVision* for those who require advanced image composition facilities. It may be used to accumulate multiple, slightly altered views of a single image to simulate such phenomena as full scene anti-aliasing, motion blur, and depth of field. This should be of particular interest to users in the Animation and Creative Graphics fields. While this feature is hardware accelerated, it is performed in "interactive time", which is adequate for all users other than those requiring real-time performance, like flight simulators.

The accumulation buffer operates in software on five span systems and in hardware on systems with ten spans. Thus, for those interested in the use of *FlexScene*, the ten span system is strongly recommended. The primary purpose of an accumulation buffer is to store images in an off-line frame buffer and allow the user to composite new images with those already stored. As stated previously, there are three major operations that can be accomplished with *FlexScene*:

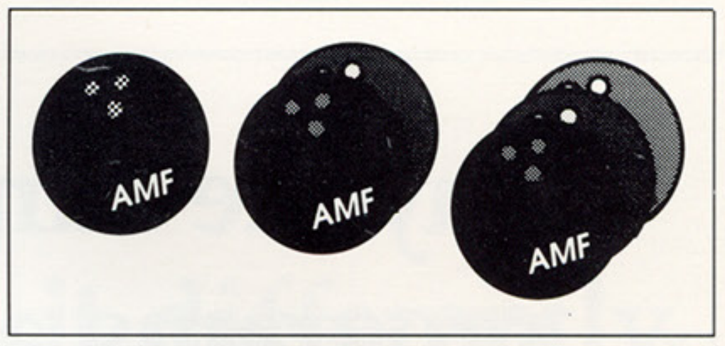
- Full Scene Anti-Aliasing
- Motion Blur
- Depth of Field Simulation

When performing full scene anti-aliasing, the same image is sent down to the *FlexScene* accumulation buffer multiple times. The more images sent, the greater the image quality. Figure 4 shows five such images. The first image to be sent is the normal, unaltered image. The subsequent images are copies of the original image *jittered* to be positioned slightly off to the right, left, top, bottom, upper right, lower right, lower left, and upper left. The combination of all of these images

slightly blurs the sharp boundaries, losing some of the sharp contrast in the image, but also eliminating most of the aliasing in the process.

Motion Blur is a similar concept, with the buffer being used to accumulate images of an object traveling along a path. Since the images being stored are all being blended together and each subsequent image is near the last, colors smear in the final image creating a blur. The reason the colors blur is that in one frame, a given pixel might be on the surface of an object, while in the subsequent frame, the object has moved to a point where that same pixel in the frame is now showing the background. The blended color for that pixel would then show some of the background through the object, effectively blurring the object along the path. Figure 5 shows an object travelling along a path, with each subsequent frame showing the accumulation of the object image in its current location together with that of all of its previous locations.

The final concept also uses the jittering of locations, but neither around a fixed point nor along a path. Depth of Field simulation may be used to emulate how a camera might have an object at the proper focal distance in focus, yet other objects in the environment out of focus. The focus of the other objects would deteriorate in some proportion to their distance from the subject. To simulate this, the sequence of images placed in the accumulation buffer must be generated by jittering the location of the eyepoint by very small distances in eight peripheral directions to the original image while leaving the objects in the environment fixed. In this manner, the images will always be sharp at the subject point of focus which is equidistant from all of the eye locations, but will blur everything else in proportion to their distance from the focal point (Figure 6).



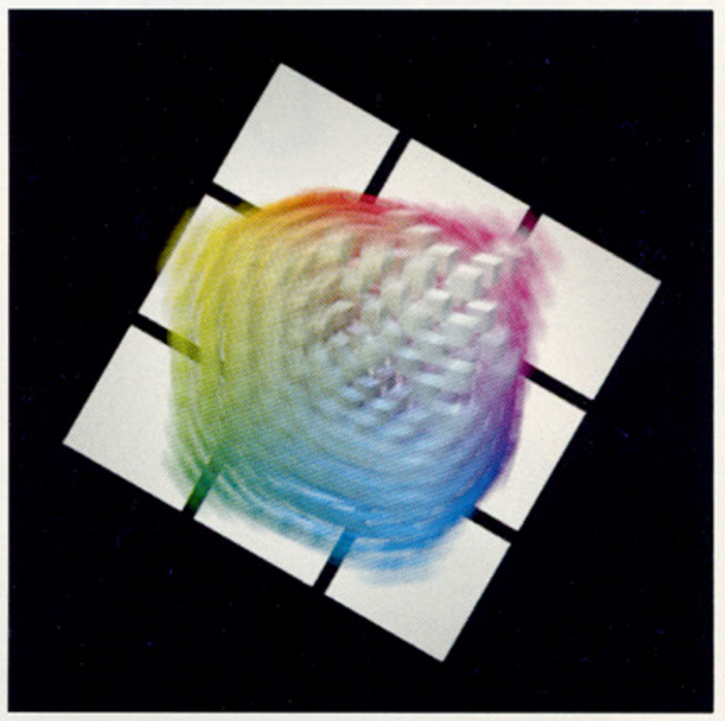
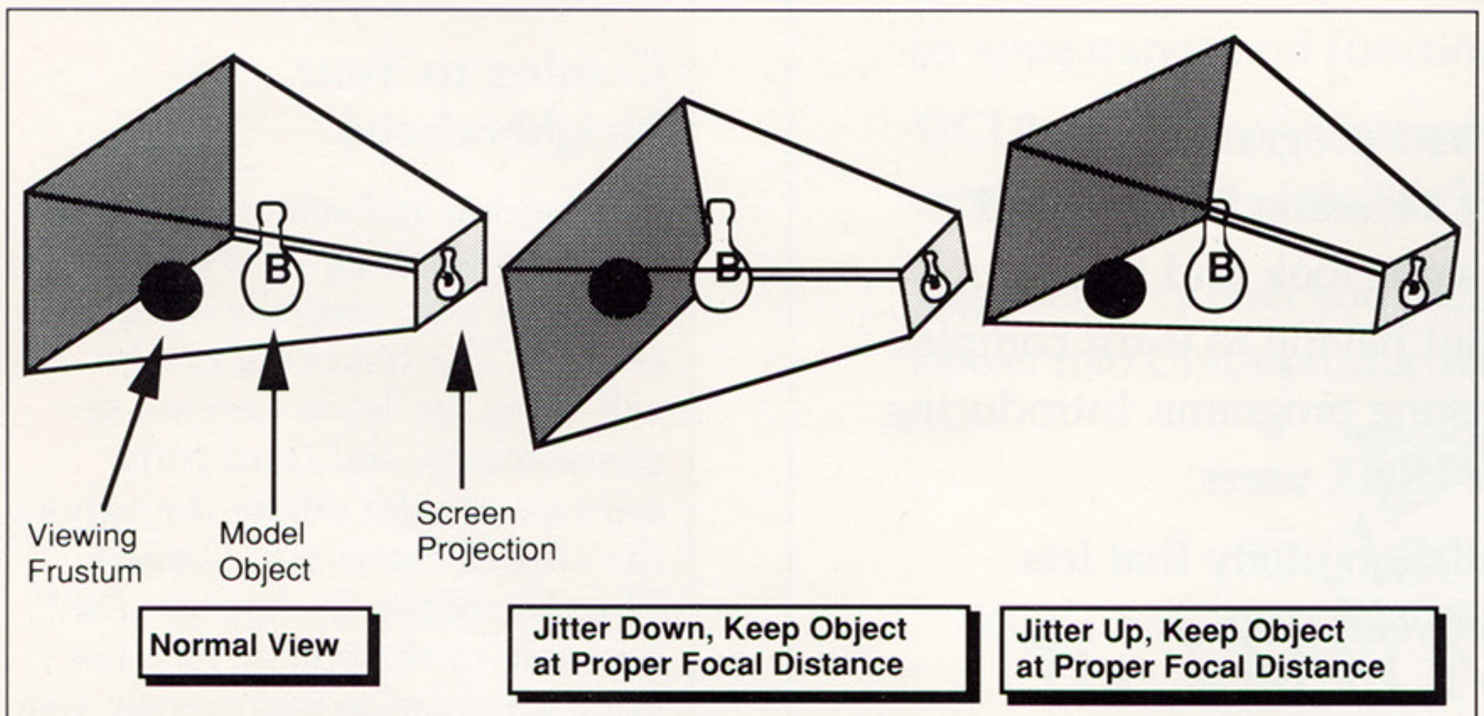


Figure 5: Motion Blur for an Object Moving Along a Path

Conclusion

Powerful features such as texture mapping, polygon anti-aliasing, and the accumulation buffer are new to the world of hardware accelerated rendering. This new level of realism and performance is the type of enabling technology which has made Silicon Graphics its reputation as the leader in graphics technology. More important even than the breadth of features or the



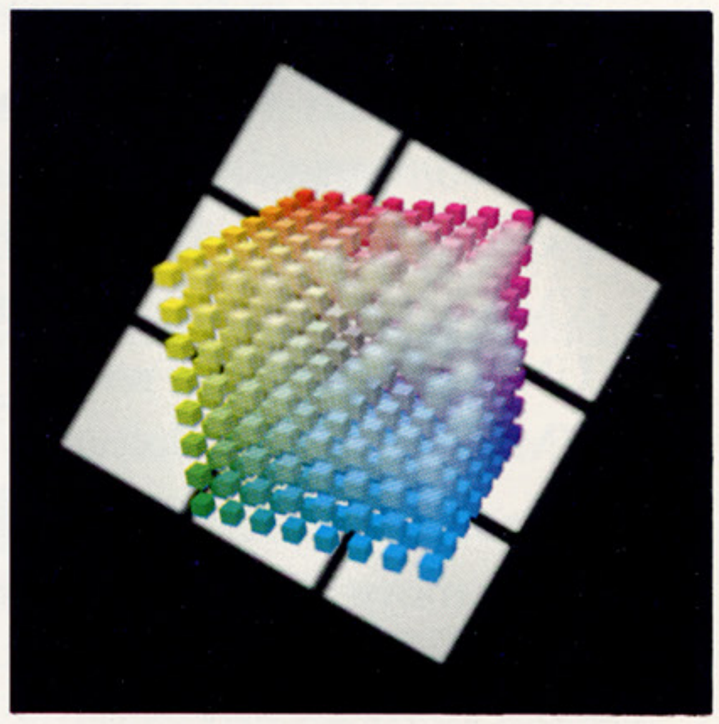


Figure 6: Jittered Views for Depth of Field Simulation

new magnitudes of performance is the achievement of these technologies' originally specified cost goals.

At a million dollars a system, this technology would be beyond the reach of most scientific and creative graphics end users. Silicon Graphics commitment to making this technology accessible dictated a system price near that of the current products. Correspondingly, this quality of rendering is, for the most part, available in one form or another in software, but the element of interactivity is lost if one has to wait 20 minutes to several hours to get a rendered frame. *PowerVision* provides the tools for envisioning reality; *affordably* and *interactively*. ■

Joshua Mogal is the PowerVision *Product Manager for Silicon Graphics.*

[1][Cla79] J.H. Clark, "A Fast Algorithm for Rendering Parametric Surfaces", supplement to *Proceedings of SIGGRAPH'79* (August 1979). [Also in Computer Graphics 13, 2, 289-299 (1979).]

Coming to Your Neighborhood

This spring and summer Silicon Graphics is taking its show on the road in a new, specially-equipped vehicle. The travelling exhibit will bring the latest systems, demonstrations, and third party software to sales offices throughout the United States and Canada. The tour begins in April and will continue to the end of July. For more information and specific tour dates, contact your local Silicon Graphics sales office.

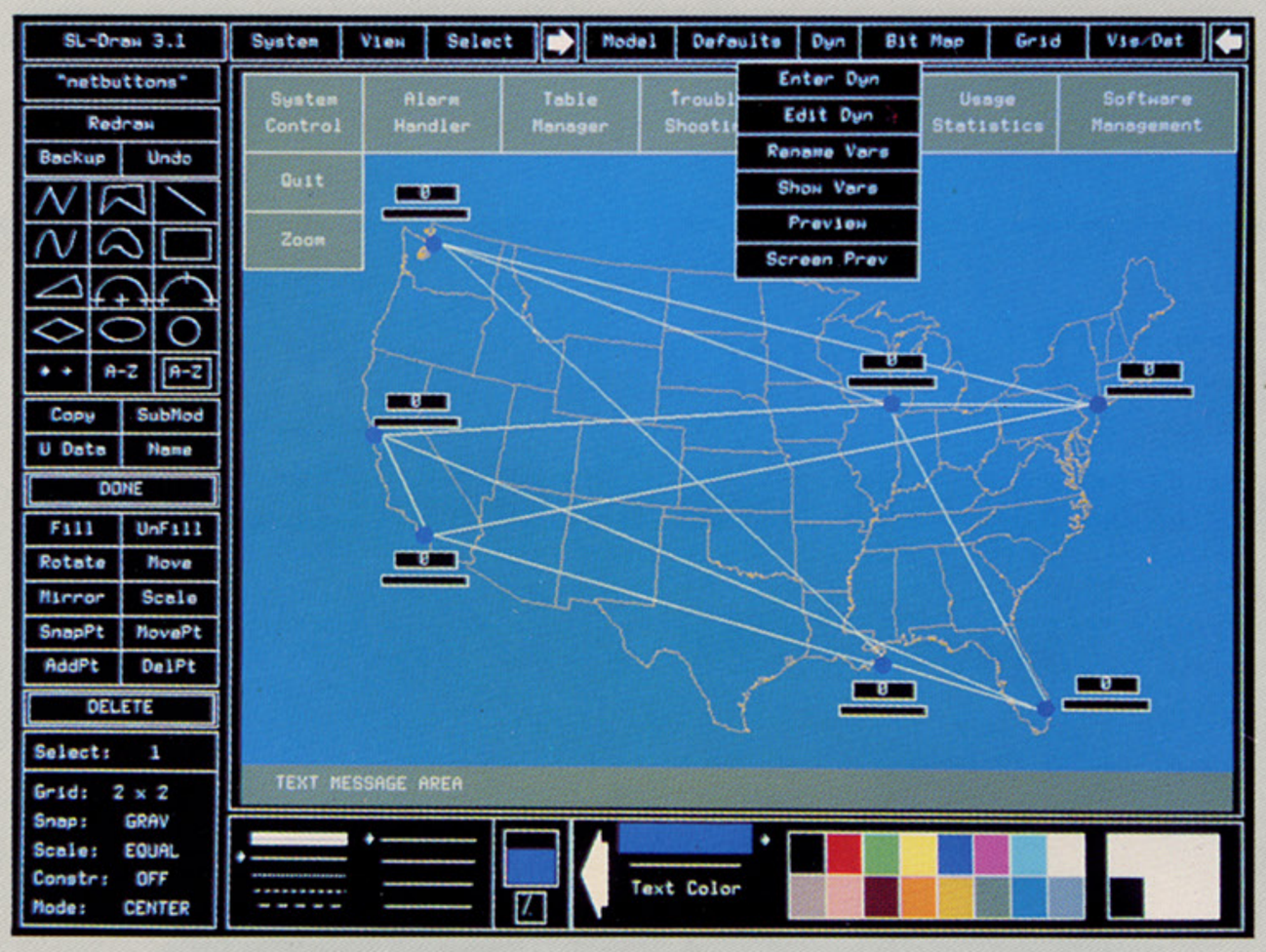
UPGRADING TO *PowerVision*

One of the major design criteria for *PowerVision* was that it should be 100 percent compatible with existing systems. This meant not only that *PowerVision* systems should be able to run existing software, but that current Silicon Graphics customers should be able to upgrade to *PowerVision* without major penalties. As a result of this design philosophy, a *PowerVision* upgrade has been made available which allows customers with installed GTX systems to remove the five board GTX graphics subsystem and install the four board Vision graphics subsystem.

This upgrade is available in two versions, one with the FX feature extension option and one without. The FX feature extension option adds memory and support for double buffered alpha memory, texture alpha, and stenciling. Once the product is installed, the user will possess a complete five span PowerVision system. For users requiring the power and throughput of ten spans from the start, an upgrade is available to bring the user from a GTX directly to a ten span PowerVision system. Note that *all* ten span systems have the FX feature extension.

The only aspect of an upgraded system that is not the same as a new-from-the-factory PowerVision system is that upgraded systems do not necessarily have a stereo-capable monitor. If your GTX system does not have a stereo monitor, and you desire to utilize the PowerVision stereo capabilities, a stereo monitor must be purchased in addition to the standard upgrade.

SL-GMS ...the complete graphical modeling system from SL Corporation.



Allows programmers and non-programmers to:

- . . . build and embed dynamic graphics screens in applications
- . . . design complex screen objects and connect them to real-time data sources
- . . . update screen objects with over 40 dynamic actions

For:

- manufacturing and process control
- network management
- avionics/cockpit display
- financial analysis and trading
- database and expert system integration

The Technology Leader in Dynamic Graphics for UNIX* and VMS Environments:

Advantages of SL-GMS include:

- A superior object-oriented architecture.
- A more powerful drawing tool.
- Direct, table-driven access to application data variables.
- More than 40 dynamic actions for graphic objects.
- Easier creation of custom graph types.
- Generally, a more open and extensible system.

A Graphics Tool that Adapts to You—Not the Other Way Around.

SL-GMS provides users with a uniquely open and extensible development system. SL Corporation believes a graphics tool should allow developers to create screens that look the way the developer wants them to look, not the way the toolmaker requires them to look. With other tools, developers wishing to go beyond vendor-supplied graphic objects and behaviors are forced to resort to raw coding. In contrast, SL-GMS not only supplies over 40 basic graph types and 40 dynamic actions, but also provides a rich set of "building blocks" that the developer can use for modifications and extensions. All SL-GMS functions can be accessed by calls to the SL-GMS function library, providing developers with additional design flexibility.

A More Powerful Drawing Tool.

Anyone can use the SL-DRAW graphics editor, a mouse-driven environment that is a powerful extension of standard drawing programs. With the point-and-click interface, users can easily create a wide variety of graphic objects and position them on the screen.

SL-DRAW allows the specification of all standard graphic attributes including color, line width, fill percent, size, rotation, position, and text font. In addition, SL-DRAW supports many CAD-like editing operations not usually found in graphics packages, including a variable-spaced grid, backup and undo functions, point congruence, snap-to-grid, pan and zoom utilities, and the ability to add to, delete from, and move the points of an object. Any graphical attribute which can be specified from SL-DRAW can be dynamically modified in response to changes in external data.

SL-DRAW derives much of its power from the object-oriented architecture of SL-GMS. The SL-GMS user can create a screen with many objects just by creating a single object (such as a meter) and "instancing" it. Dynamics can be applied to the "generic" object and/or to the instances themselves. For example, some attributes can be attached to all instances of the object at once, while a different property or dynamic behavior can be specified for each instance. The result of this hierarchical, object-oriented approach is an increase in developer productivity that cannot be matched by "flat" systems.

*Including 386-based systems.

Output Dynamics

The dynamics functions used to animate screen objects can be specified from the SL-DRAW editor. These functions establish direct connections between screen elements and application database variables. Screen objects and object components—even sub-component elements and text—can be animated to reflect real-time changes in application variables.

Direct Access to Application Data

The advanced architecture of SL-GMS makes it possible to control animation and dynamics through a simple table-driven approach which links data variables to screen elements.

Input Dynamics

Screen objects created with SL-GMS can also be used by the end-user to interact with the application. SL-GMS "widgets" can perform actions, evaluate expressions, reference variables and call user-defined functions, as well as input data values and switch between screen states.

Text Editor Option

The SL-GML Language interpreter is a full-command alternative/complement for editing screens in text mode. It handles conversion of binary-screens to ASCII or C-structure files for portability across platforms or compilation into diskless runtimes. It can make de-bug format dumps of screen files. SL-GMS simplifies the layout of complex screens which involve multiple, tiled, or overlapping views.

Advanced Technology

The design of SL-GMS is founded on the SL-Object Oriented Environment (SL-OOE)—a tested, stable and pioneering implementation written in straight C. This kernel environment is complete, simple and elegant, and requires no special compilers, pre-compilers or C-language extensions, even though users wishing to integrate with C++ or Objective C may do so.

Designed Portability Protects Your Investment

Screens and programs built with SL-GMS on any supported platform run automatically on any other supported platform. This allows users to build screens on lower-priced, lower-performance hardware, and then run them on faster, more capable hardware. . . or the other way around. Screens you have developed will remain transparently functional despite technical change, because SL-GMS is truly device-independent.

Licensees of SL-GMS Include:

The Boeing Company
Chrysler Motor Corporation
Combustion Engineering
Cummins Engine Company
Digital Equipment Corporation
E.I. DuPont
Eaton Corporation
Eurotherm Corporation
Fokker Space & Systems
General Electric Company
Hewlett-Packard
Hughes Aircraft Company
Johns Hopkins University, APL
Lawrence Livermore National Labs
Lockheed
The MITRE Corporation
Martin Marietta
Northrop Corporation
Salomon Brothers, Inc.
Westinghouse Nuclear Division

License Prices

Development Configuration	
SL-GMS-dev	$15,500
Run-Time Module	
SL-GMS-rt	$1,800

Call or write today for additional quantity, distributor or VAR discounts: 415/927-1724, FAX 415/927-2931.

SL Corporation
Suite 110 Hunt Plaza
240 Tamal Vista Boulevard
Corte Madera, California 94925
formerly
Sherrill-Lubinski Corporation

THE RIGHT TOOLS FOR THE TASK

Using computers ranging from PCs to Crays, the author investigates the advantages and disadvantages of different methods utilized in 3D visual simulations.

BY LAURENCE A. FELDMAN

Introduction

The objective of 3D computer simulation is to mimic the behavior of real world phenomena at a cost significantly less than that of performing an actual experiment. Using the sophisticated features of computer graphics, the simulation can visually approach the real world experiment. Adding graphic realism not only aids in analyzing the results but adds a degree of credibility (sometimes undeserved) to the simulation process. However, to effectively utilize computer graphics one must examine the computer environment (i.e. cpu speed, graphics processing speed, data transfer rates, memory capacity, disk capacity, and algorithm speed) which is required in order to achieve maximum performance.

The application employed for the purposes of this article was 3D simulation of the air flow over an arbitrary aerodynamic body (an airplane). The computers used were PCs (a Dell 386, a Macintosh), workstations (an IRIS-4D and a Sun-4), and supercomputers (a Cray-2 and a Cray-YMP); the networks were ethernet and ULTRA. Graphics devices included: PCs, workstations, and the ULTRA frame buffer. The Kirtland Air Force Supercomputer Center in Albuquerque, New Mexico, operated under the direction of Dr. N. L. Rapagnani, served as the principal site for these investigations.

Performance Criteria (CASE)

This article focuses on the four criteria used to evaluate the effectiveness of 3D simulation: *Cost*, *Accuracy*, *Speed*, and *Ease* of use (*CASE*).

Accuracy relates to how well the simulation approximates reality. If the flow solution and wind tunnel results depart significantly, then what good is the simulation?

Cost concerns not only the cost of computers, graphics, and networking, but the expense of computer time and person-hours in performing the simulation. The simulation cost must be significantly less than a wind-tunnel test to justify its use.

Speed is measured not only by the compute time of simulation, post processing, graphics analysis, and data transfer rates, but more importantly, by the time it takes to complete one entire simulation.

Finally, *Ease* of use pertains to how simple or difficult it is for the entire system to perform the simulation, transfer the data, analyze the results, and redesign for the next simulation. Is the system easy to learn, easy to use, and not subject to breakdowns of integral components?

An ideal simulation *CASE* is one that predicts phenomena very close to the results of a real world experiment at a negligible cost, and at speeds at which "What If?" scenarios are feasible. The simulation is further enhanced if the individual running the experiment is entertained by the system rather than plagued by it.

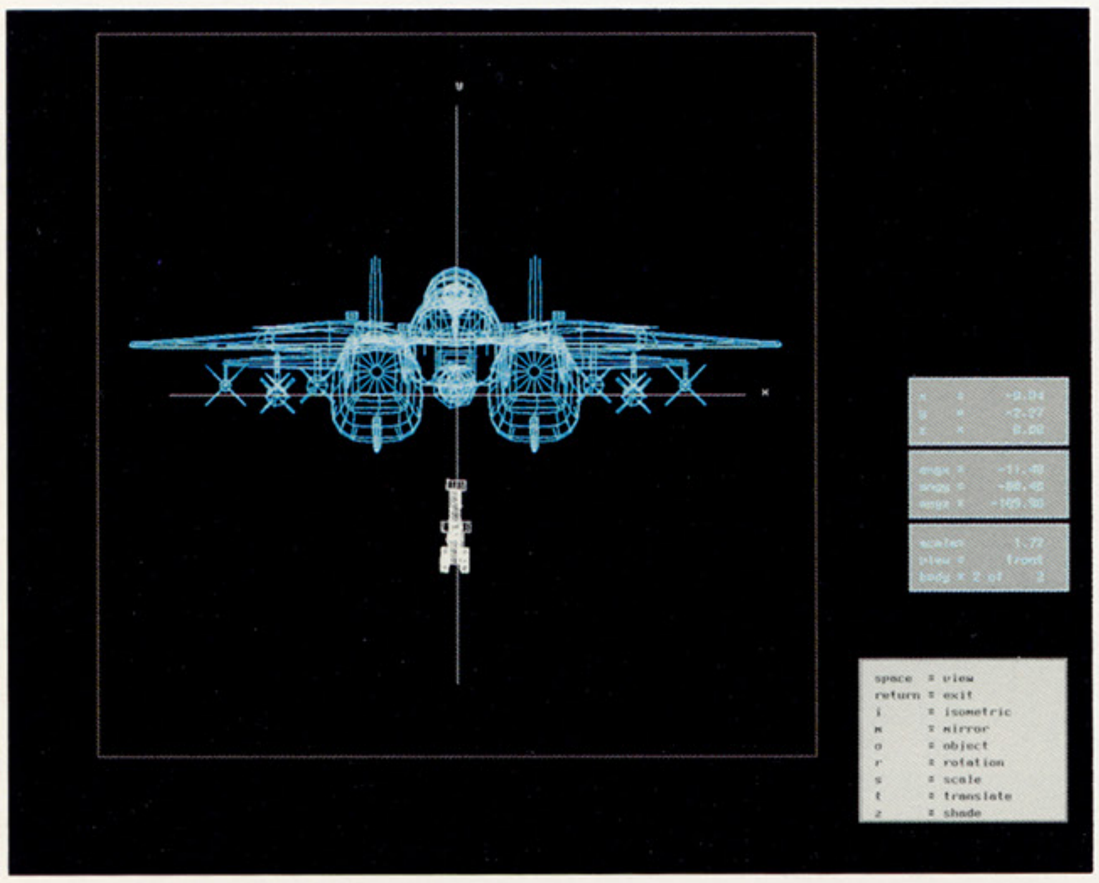


Figure 1: Each airplane is composed of simple components (primitives) which are derived from spheres, cones, etc. A toy airplane is used as a basis for defining the fuselage, canopy, missiles, etc. Simple dimensions such as length, radius, and angles are measured on the toy airplane and used to define the geometry of the primitive. Primitives are then "glued" to form more complex objects. This approach avoids the need to measure the geometry over the entire surface of the airplane (3D digitization). Figure 1 illustrates the "glueing" of landing gear to an F-14 by REPLICORE, performed interactively using the mouse. The IRIS-4D graphics processors allow for "real-time" manipulation.

Three Stages to Simulation

The first stage, *Pre-Processing*, consists of geometrically modeling the airplane and defining a finite grid or mesh around the plane's exterior.

The second step, *Processing*, solves the non-linear partial differential equations describing the flow around the airplane's body. The flow solution consists of computing density, velocity, and energy values known at pre-defined locations (mesh points) as a function of time. Using these parameters, lift, drag, moments, and other aerodynamic properties are evaluated and used to predict the performance of the design under a variety of flight conditions, speed, pitch, and so forth.

The last step is the performance of *Post-Processing* — visual analysis of the integrated solution — to observe aerodynamic behavior of the body. This refers to the observation of the flow patterns, shock structures, positions, pressure coefficients, and other aerodynamic activity.

The goal of this Computational Fluid Dynamics simulation is to gain insight into the behavior of flow around an aerodynamic design and reduce or eliminate the need for expensive wind-tunnel testing.

Issue No. 1: What is the Best Type of Host Computer?

The ideal computer would be one that has fast computing power (gigaflops), such as the Cray-YMP, a large memory, say 512 megawords (4 gigabytes) like the Cray-2, and a rich library of 2D and 3D graphics routines complementing high-

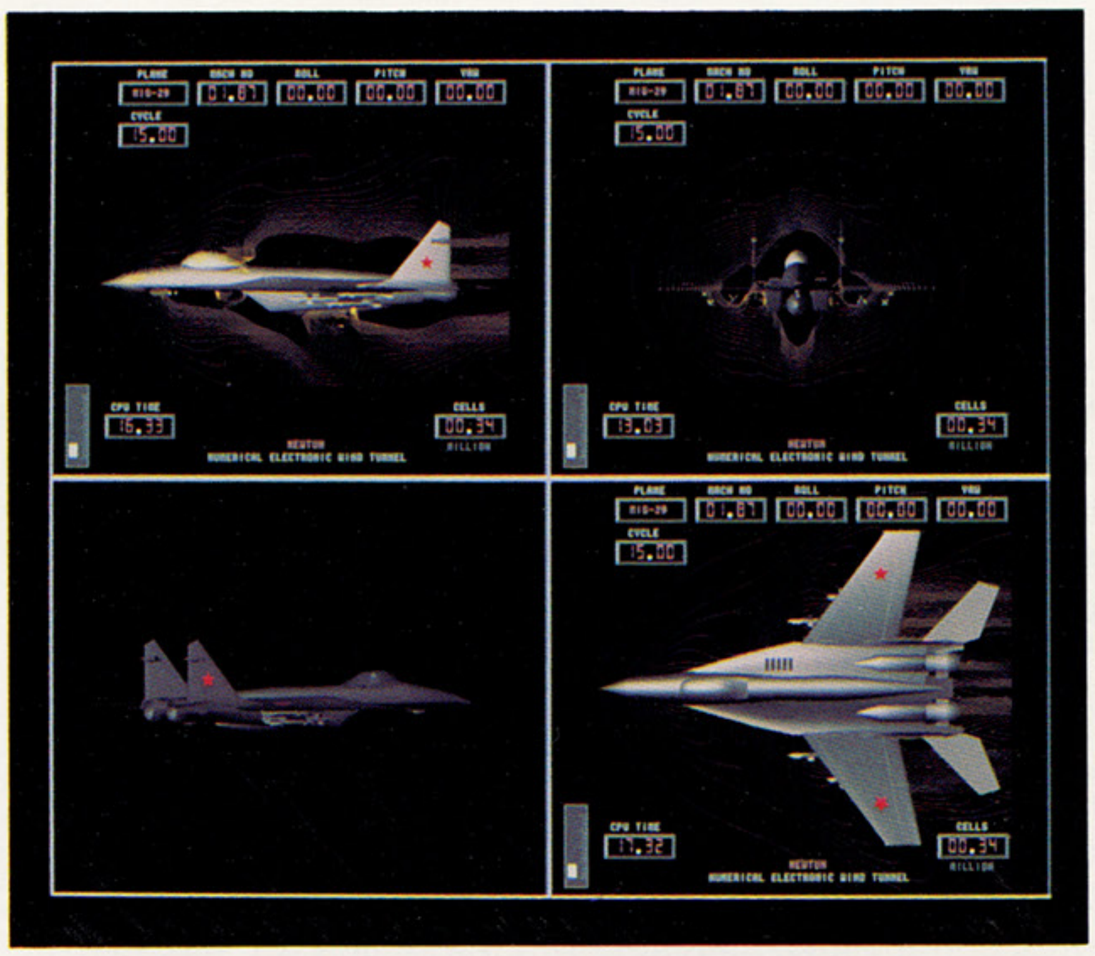


Figure 3: Four views of a Mig-29. The contour lines represent constant Mach Numbers. NEWTUN calculates the transient behavior of flow over bodies in a uniform grid. The plane's orientation and flight speed are defined by the user at time zero and the flow is then computed at discrete intervals over a pre-defined grid.

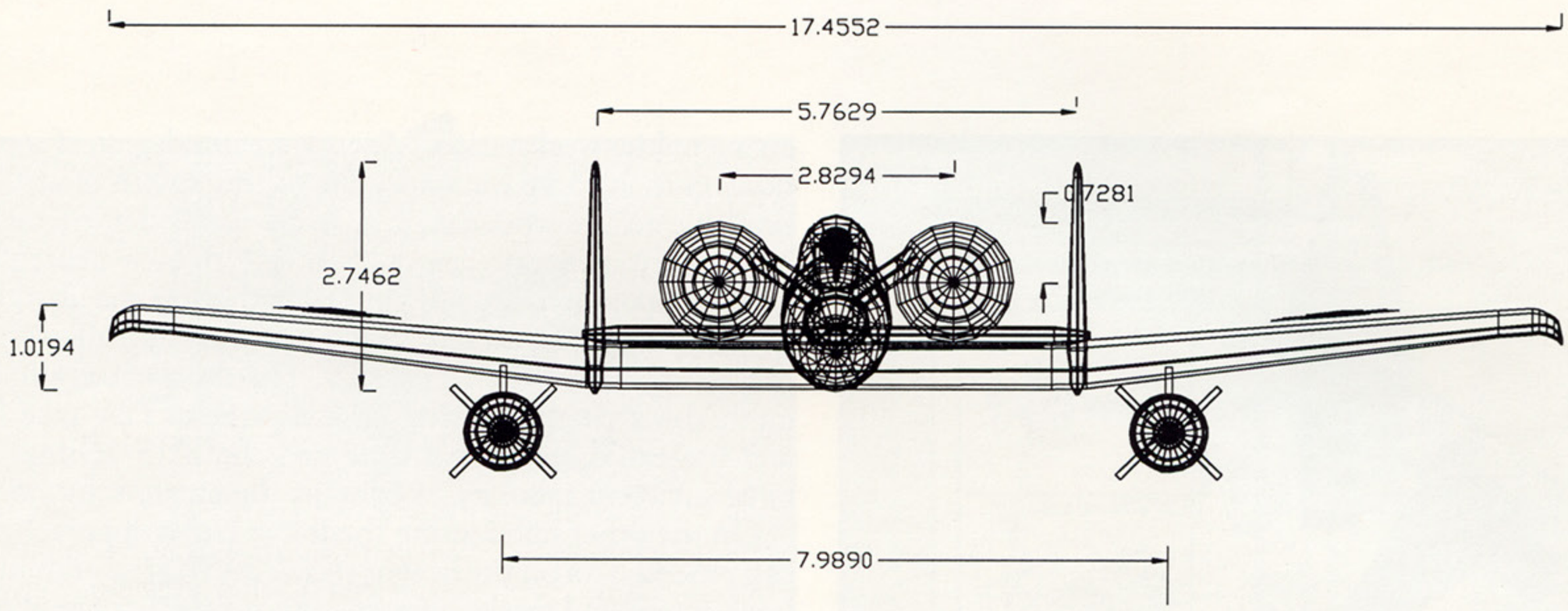


Figure 2: The plane is represented by a polygonal data base produced by programs such as AUTOCAD, MOVIE-BYU, and others. AUTOCAD incorporated REPLICORE data and performed tasks including interactive dimensioning, shading, and animation, on PCs (an IBM, a Macintosh, a Dell 386) and workstations. Figure 2 illustrates a front view of an A-10

speed graphics processors, the IRIS-4D Series, for example, all at the cost of a personal computer. Unfortunately, no such computer exists, so tradeoffs are necessary.

The Cray computers possess two principal advantages: speed and memory. However, third party software is limited and cost is prohibitive. Only the privileged have local access to Crays and most remote users are burdened with slow data transmission. No generic graphics library exists and all 3D processing is done by user supplied high-level graphics code. Graphics images must be displayed on remote terminals, workstations, or frame buffers where speeds are regulated by data transfer rates which are typically slow.

Super-minicomputers (Convex-240) provide approximately 1/10 to 1/40 the speed of the powerful Cray computers at about 1/20 the price.

A high-end graphics workstation (possessing 128 megabytes memory, capable of parallel processing, and with compute speeds of approximately 10 to 40 megaflops) such as the IRIS 4D 280 costs in the $180, 000 to $300, 000 price range and can tackle multimillion cell grids at speeds 1/100th that of a Cray YMP. More importantly the pre-processing, simulation, and post processing reside on one computer, thus (1) eliminating the need for networking large data bases, (2) expediting flow visualization, and (3) performing interactive visual anaylsis concurrent with the ongoing simulation.

Issue No. 2: How Important is a High-speed Network?

Perhaps the leading bottleneck in quickly completing simulations is the speed at which data is transferred from computer to computer. If one is running a simulation off-site and creates a flow-field solution represented by 10 to 100 megabytes or more of data, it can take 6 to 24 hours to download this information to a workstation for post-processing analysis.

When the possibility of equipment failure during transmission is taken into account, the task becomes even more agonizing. If high-speed networks are not available, one alternative is to perform the post-processing on the host computer, downloading compressed images to the workstation (see Issues 3 and 5). If on the other hand , one is running the simulation locally and wishes to visualize results spontaneously, a high-speed frame buffer provides the solution. (see Issue 3). The problem of slow inter-computer movement of information is most easily solved by eliminating the need for data transmission. This is possible only when the same machine is used for both the simulation and to generate graphics. Computers like the IRIS Power Series are a step in that direction but, as discussed above, cpu speed, word-size, and memory must be able to undertake million cell grids in a "reasonable" length of time.

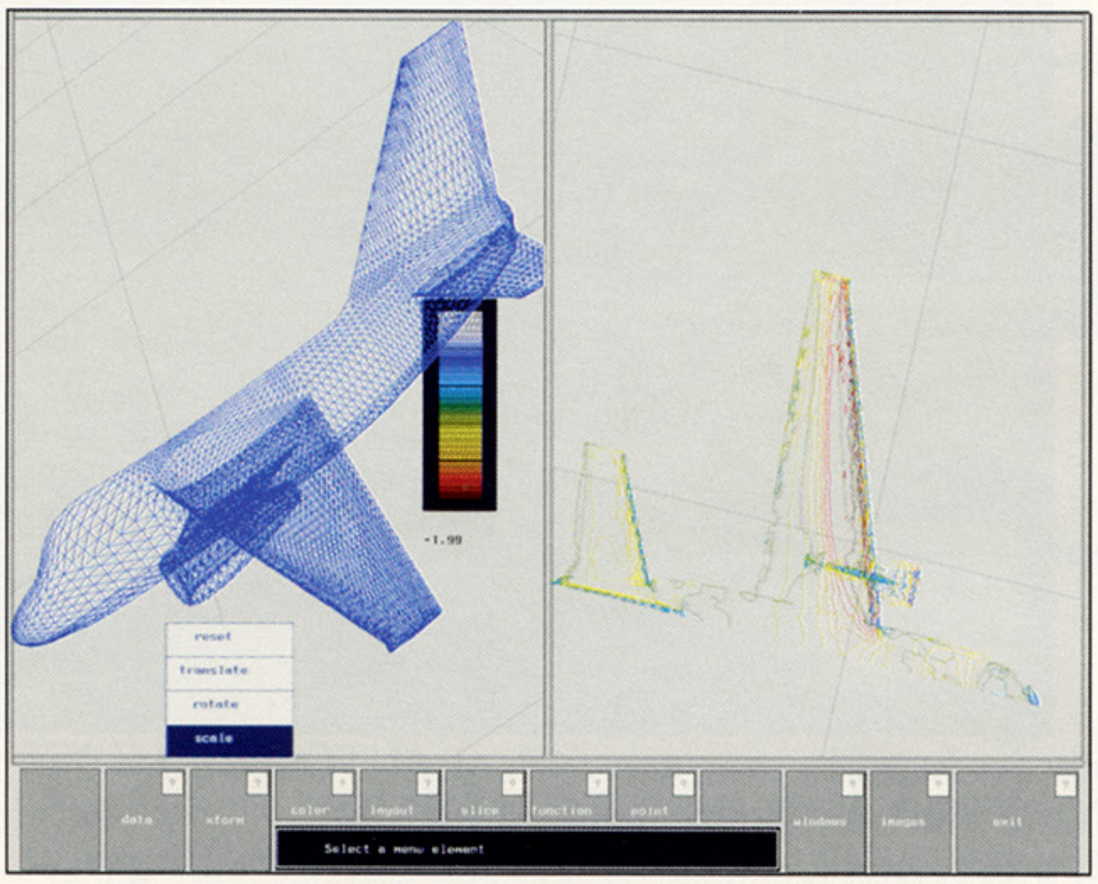


Figure 4: SHO_FLO employs an interactive menu system for post-processing graphics. The entire flow-field (density, three velocity components, and energy) are stored in memory. The user may specify a region where contouring, tracer particles, carpet plots, or 3D arrows are displayed. The user can then plot any of 58 parameters as defined by NASA Ames PLOT3D, rotating the results in real-time using the IRIS-4D graphics processors. Figure 4 illustrates lines of constant pressure coefficient generated from the AIRPLANE flow solver (Princeton University, Antony Jameson).

Issue No. 3: Frame Buffers vs. Interactive Graphics Workstations

There are two basic approaches to performing post-processing analysis for computational fluid dynamic simulation. The first is to download or transfer the flow-field solution to a workstation then visualize the results using the graphics engine and graphics library of that system, e.g. the IRIS-4D. The second approach calls for performance of the post-processing analysis on a host computer and downloading the resulting images to a frame buffer (terminal, PC, workstation, etc.). Each method has advantages and disadvantages; actually the two approaches complement each other.

The advantage to the former approach is that the graphics on a graphics workstation possess a greater degree of sophistication than those on a host due to the wealth of software available to the workstation user. Secondly, an interactive interface is generally more desirable than a "batch" or "script" approach. In addition, distributing the workload frees the host computer for simulation tasks and allows post-processing to be done in parallel. The disadvantage to this method is that transferring large flow-fields may take long and frustrating periods of time and the IRIS-4D may not have sufficient memory to contain the entire solution.

On the other hand, using the frame buffer approach, one can process data on the host at speeds much greater than the workstation and produce image output files that are of fixed size regardless of the grid resolution. Thus, the transfer rate is faster and more manageable. One disadvantage in this case is that graphics software is non-existent on the host and must be supplied by the user. Another shortcoming is that images are defined prior to simulation disallowing more spontaneous investigation through interactive evaluation.

A compromise approach is the porting of a graphics library to the host computer and downloading "display lists" to the workstation.

Issue No. 4: Do I Really Need to Multi-task and Vectorize my Code?

Multi-tasking is a procedure in which system calls are added to a program in order to use multiple processors in parallel (IRIS Power Series, Cray). Using vectorization, scalar operations are made into vector operations in order to take advantage of vector capabilities of certain computers (Cray series).

The original NEWTUN, partially vectorized and employing only one processor, ran at a speed of 50 megaflops on a Cray-YMP. When the compute-bound section of the program was fully vectorized and multi-tasked for eight processors, the same Cray-YMP achieved floating point speeds in excess of 1.2 gigaflops (B. Dodd/ Cray Research Inc., Albuquerque). Vectorization improved the graphics speed by nearly an order of magnitude. Vectorization techniques were found to improve the performance of codes running on non-vector machines such as the Sun-4 workstation or IRIS-4D Series.

Both procedures may require weeks or even months to program but the benefits are obvious in terms of speed and cost. The accuracy of the simulation is also improved since lower cost and faster processing translate to a more refined grid resulting in fewer errors in solution.

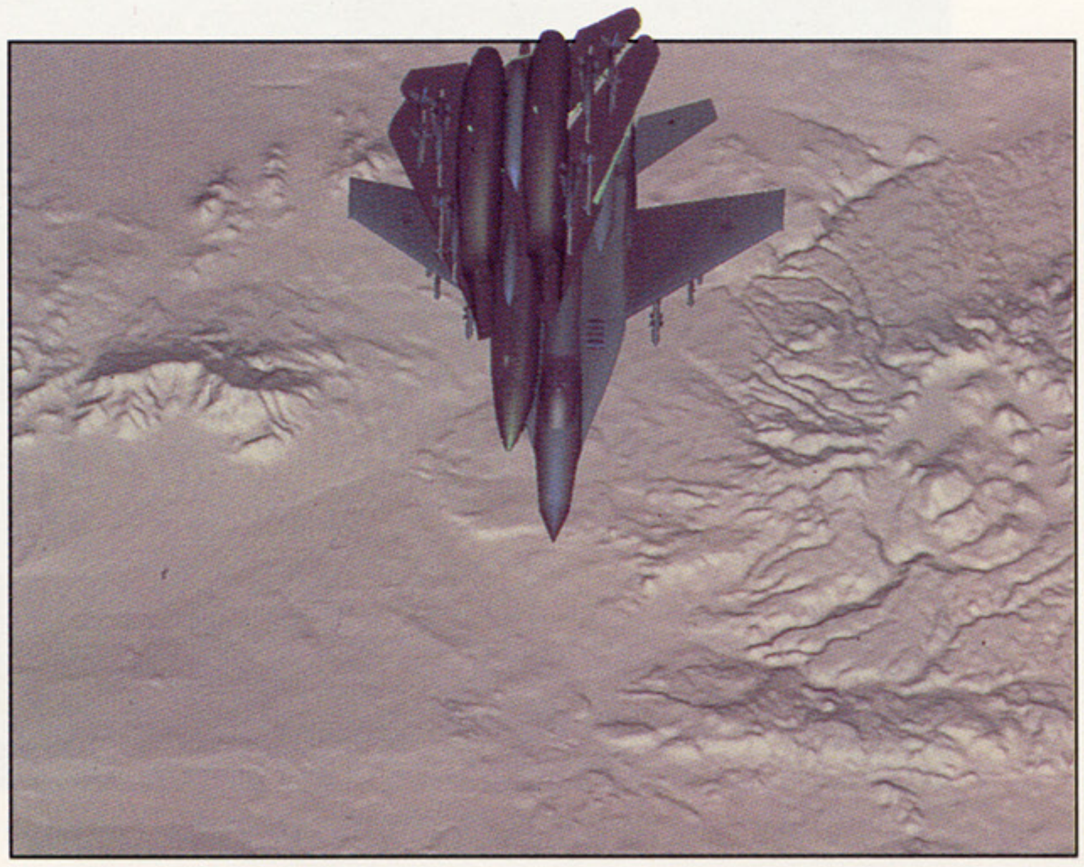


Figure 5: The image depicts a 60 square mile quadrangle of Albuquerque, New Mexico (data supplied by US Geological Survey, Menlo Park, Ca.) with an F-14 and Mig-29 in Top Gun scenario. REP_TILE produced the 200,000 polygon image on the Cray-2/ Ultra Frame Buffer in approximately 1-3 seconds. REPLAY then played back the scenes in slow, medium, or fast mode (depending on available memory) and allowed for forward, reverse, stop, and start. REPLAY also allowed for multiple scenes to be catenated and/or dissolved. The rate of replay was 6 to 24 frames a second depending on mode and screen resolution. The ULTRA frame buffer consistently achieved transfer rates approaching 100 megabytes per second. REPLAY runs on the Cray-XMY, Cray-2 and Cray-YMP. REPLAY was authored by B. Dodd/Cray Research, Inc.

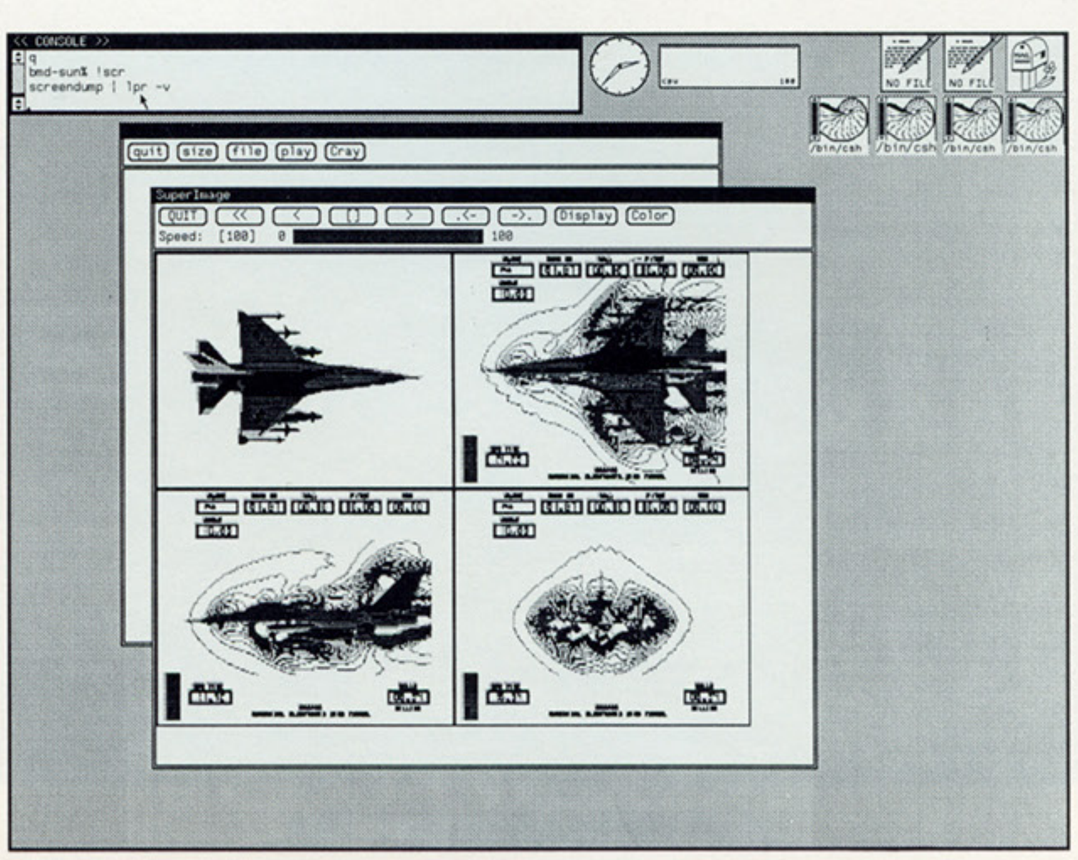


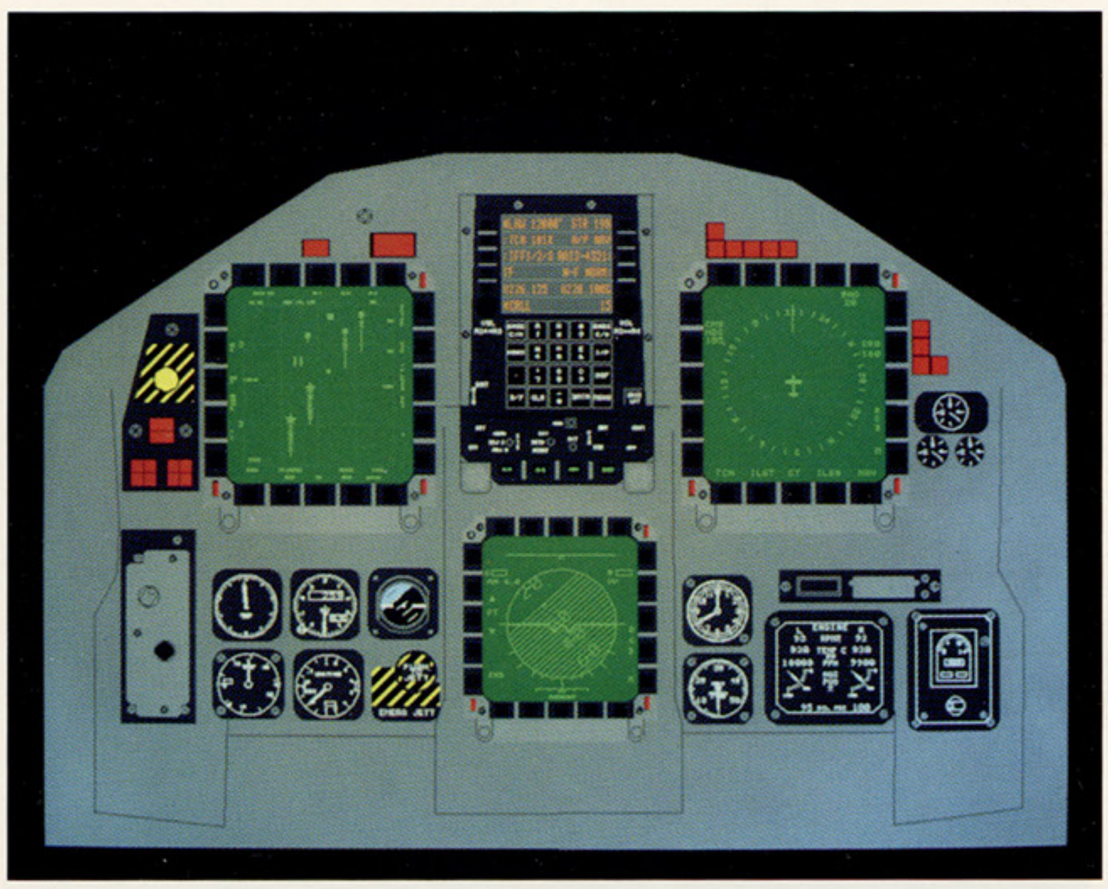
Figure 6: SUPER IMAGE is a distributed raster animation program which runs in a workstation window environment and communicates with a remote application via TCP/IP. The workstation interface consists of multiple windows, panels, buttons, and sliders. The interface transmits messages through mouse or keyboard i/o. Concurrent simulation can be displayed and manipulated via user input and recorded, if the user wishes, for later playback. Graphics are displayed on black and white or color terminals. The program was developed by Darragh Nagle and Brian Dodd of Cray Research Inc. Figure 6 shows four views of an F-16. The contours represent lines of constant Mach Number for a free stream Mach Number of 1.86.

Issue No. 5: How Beneficial is Compression of Images and Data?

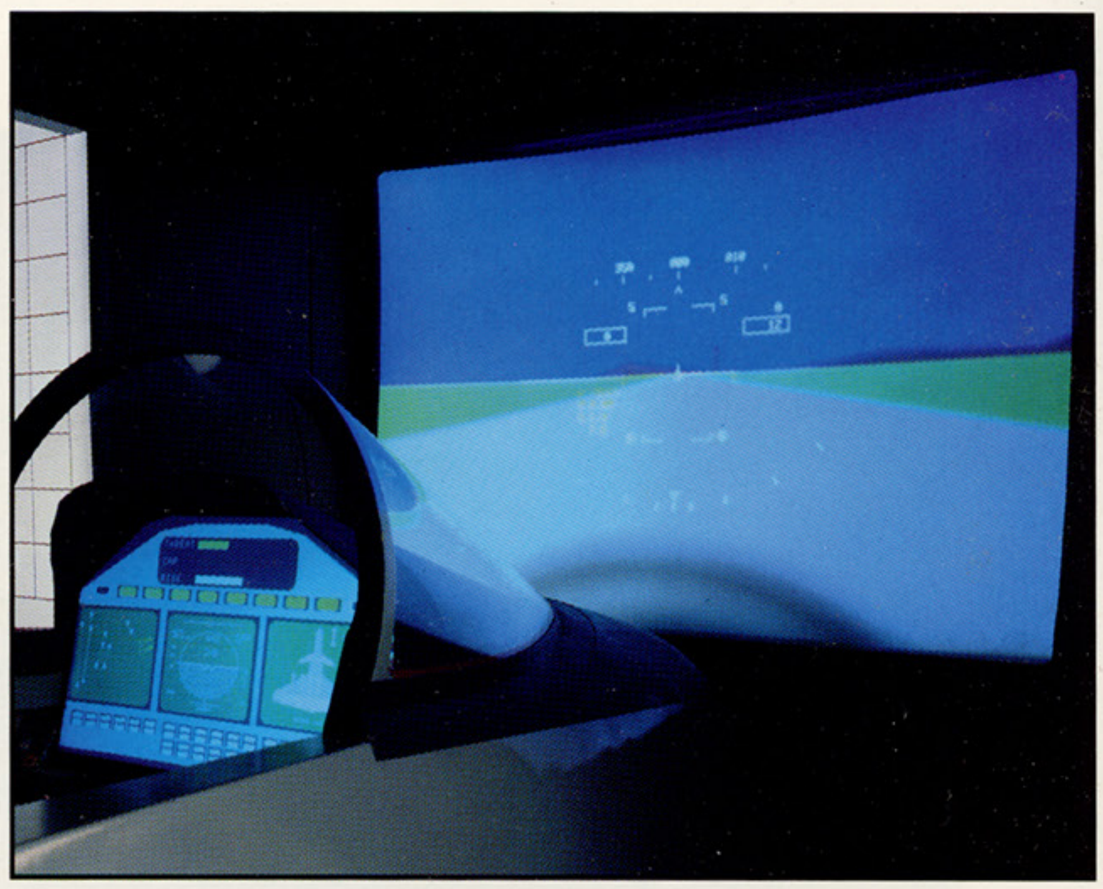
When performing 3D simulation on the Cray-2, images were generated at fixed intervals and simultaneously displayed and stored on the disk. After the simulation was complete, the stored images could then be used for "replay" at rates of 24 frames per second on the Cray-2/Ultra frame buffer. A REPLAY program was written to display the image files in a manner similar to that of a video recorder. Additional features, such as the ability to dissolve or delete images, were added to enable post-processing editing.

Since 120 frames of 1280 x 1024 24 bit images can require large disk allocation, compression (and expansion) routines were developed to reduce the size of these files. In animation scenarios it was also discovered that if one subtracted image (n) from image (n+1) and stored the result using a rectangular compression algorithm (Weapon's Laboratory/Gleisher-Conley), the original image file could be reduced, in most cases, by 99 percent. More important than just reducing file size to save disk space (which was the initial motivation), it was found that the reduced files could be easily transported over networks to other graphics devices (a Macintosh, a PC, a Sun, an IRIS) and decompressed on these remote computers for real-time display. Although the other graphic devices typically offered only 8 bits of color, it was found that

What do these applications

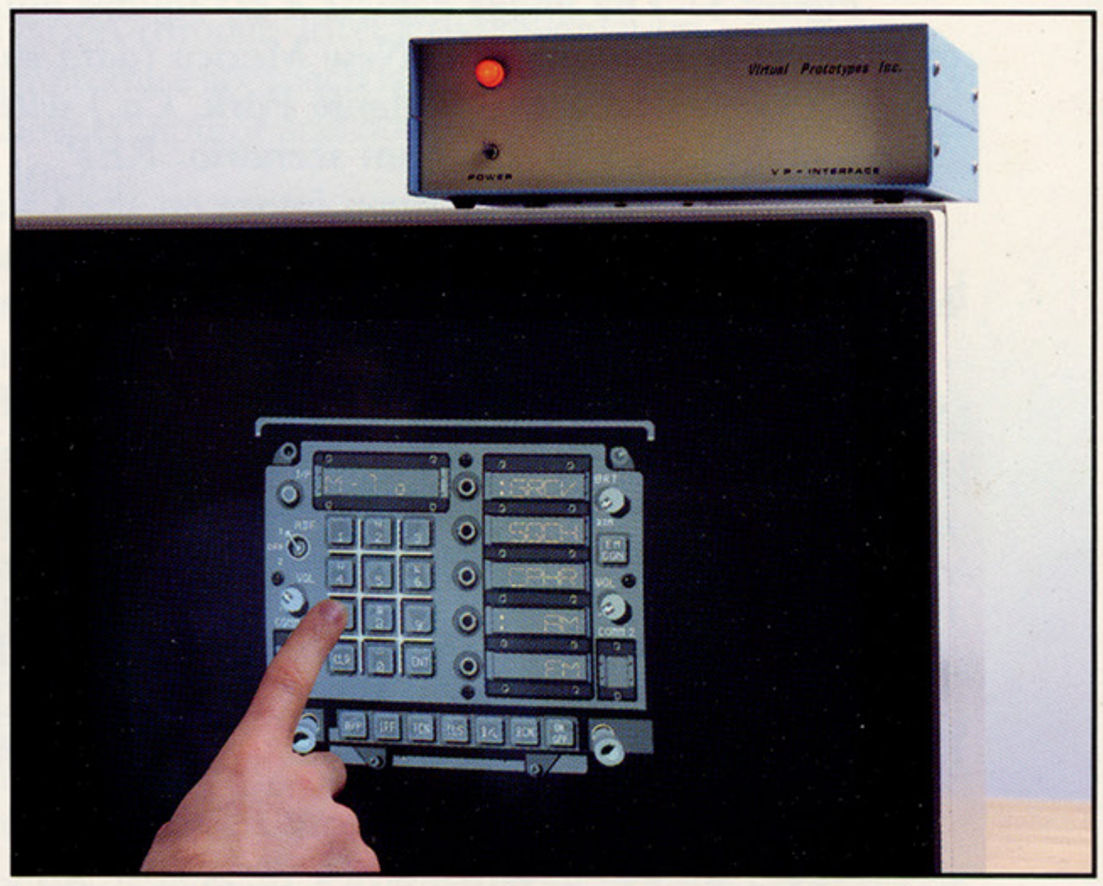


AEROSPACE



FLIGHT SIMULATION

AVIONICS



TRAINING

They are fully functional, interactive and work in real time.

They were created 10 times faster than possible with conventional technology.

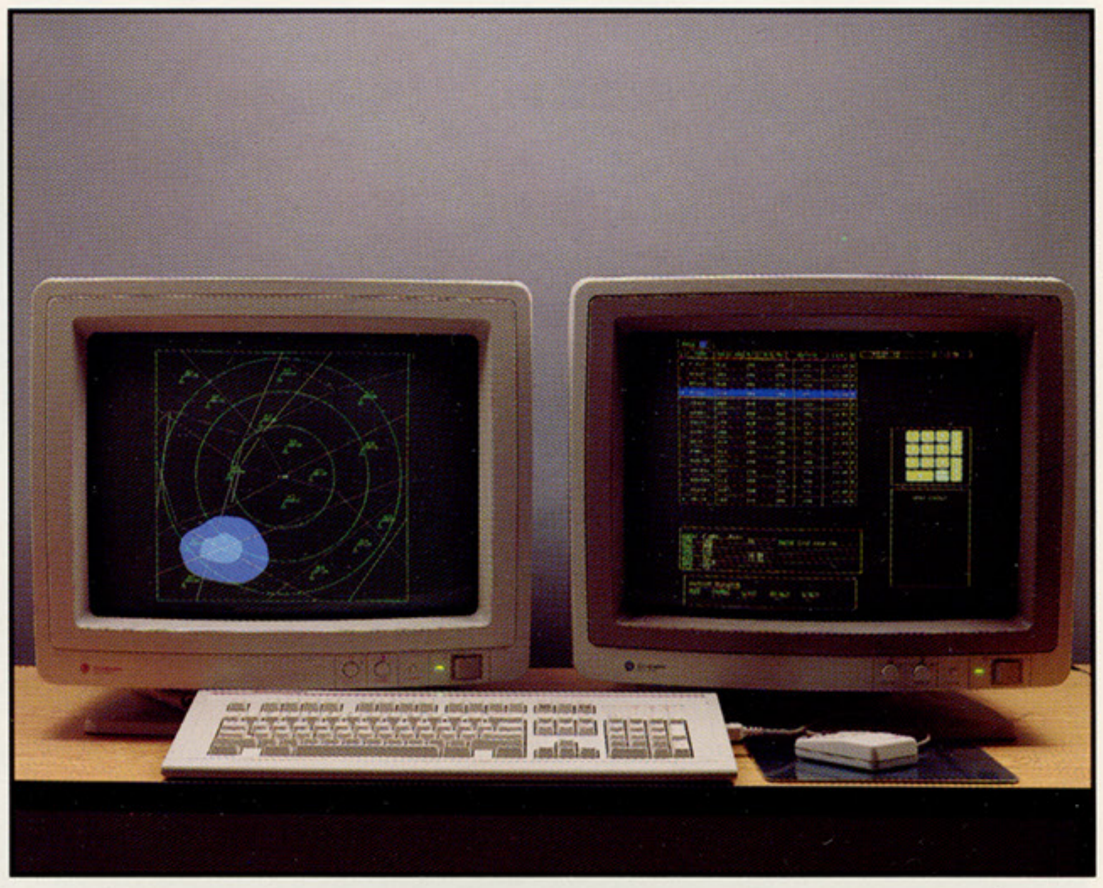
The breakthrough technology for:

- **Rapid Prototyping**
- **Simulation and Training**
- **Automatic Code Generation**

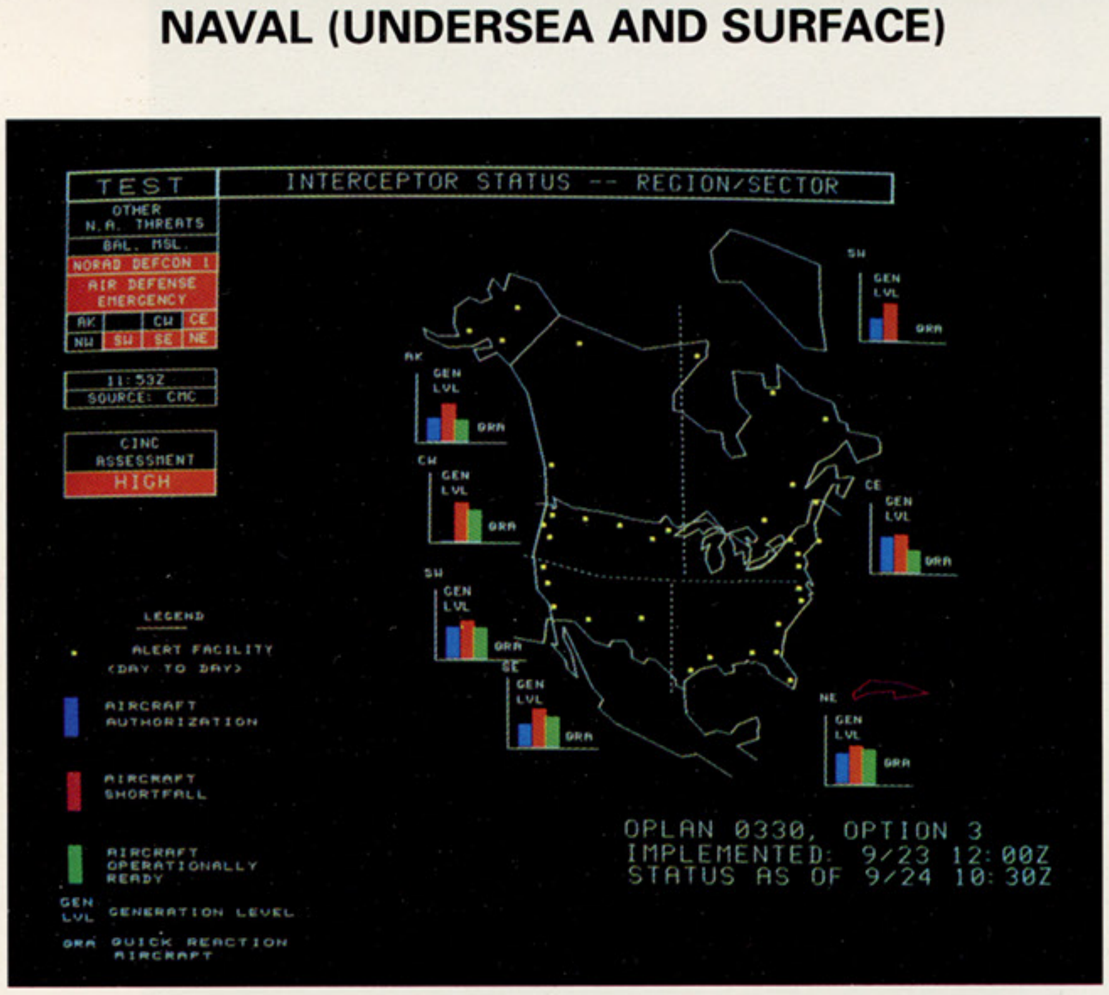
They were created with VAPS.

have in common?

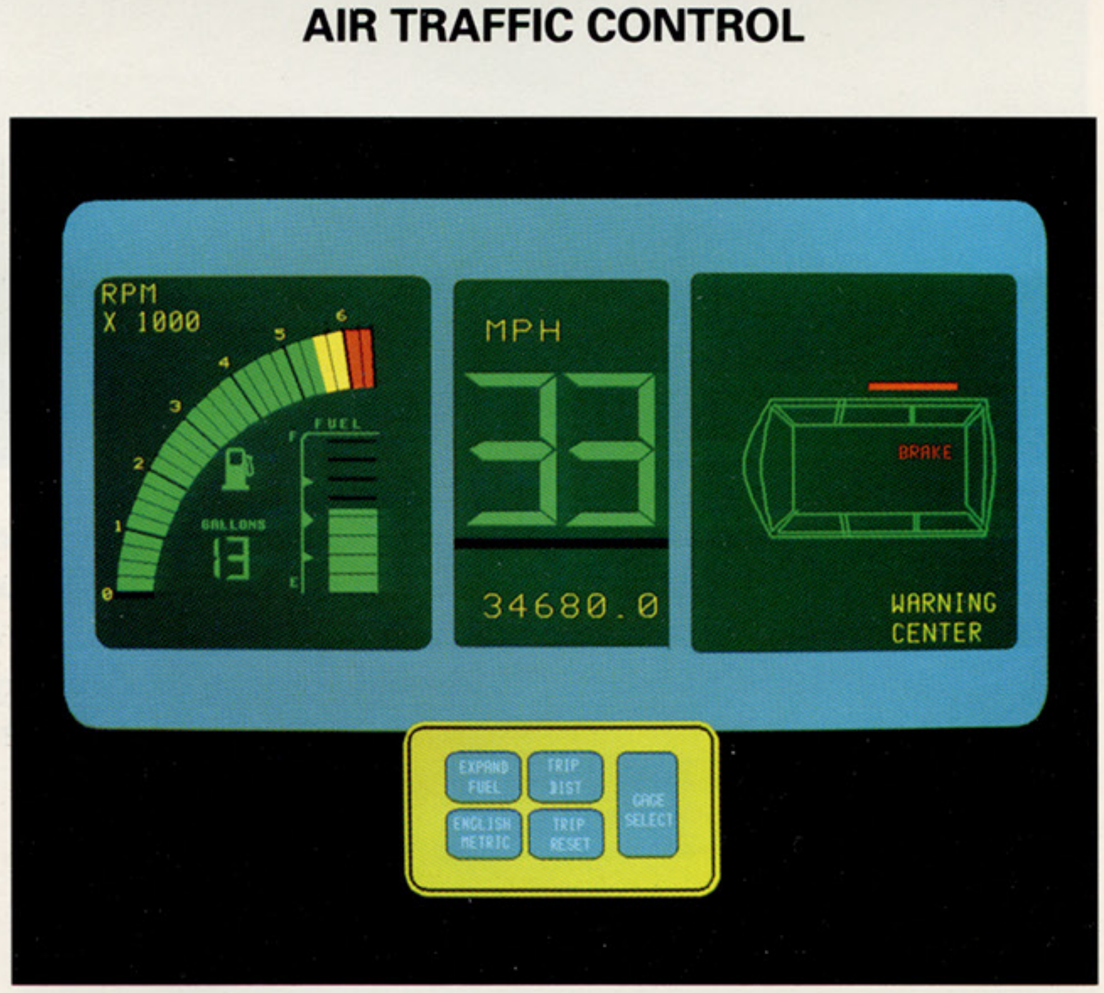
NAVAL (UNDERSEA AND SURFACE)



AIR TRAFFIC CONTROL



C³I



AUTOMOTIVE

Virtual Prototypes Inc.

5252 de Maisonneuve West, Suite 318,
Montreal, Quebec, Canada H4A 3S5

Tel.: (514) 483-4712
Fax: (514) 483-1231

France:
Métrologie S.A.
La Tour d' Asnières
4, avenue Laurent Cély
92606 Asnières Cedex, FRANCE
Phone: (33) 1-40-86-82-62
Fax: (33) 1-47-90-84-02

West Germany:
GEI Rechnersysteme GmbH
Roermonder Straße 615
D-5100 Aachen
WEST GERMANY
Phone: 0241-17-10-81
Fax: 0241-173267

Japan:
C. Itoh & Co. Ltd
2-5-8 Kita-Aoyama
Minato-ku Tokyo, Japan
Phone: (03) 497-8291
Fax: (03) 497-8300

United Kingdom:
Thame Microsystems Ltd.
Thame Park Road
Thame Oxfordshire OX9 3UQ
Phone: 084-426-1456
Fax: 084-426-1682

Value Added Reseller
Geometry Partners Program

SiliconGraphics
Computer Systems

CANADA AWARDS FOR BUSINESS EXCELLENCE
Gold Award 1988 Innovation

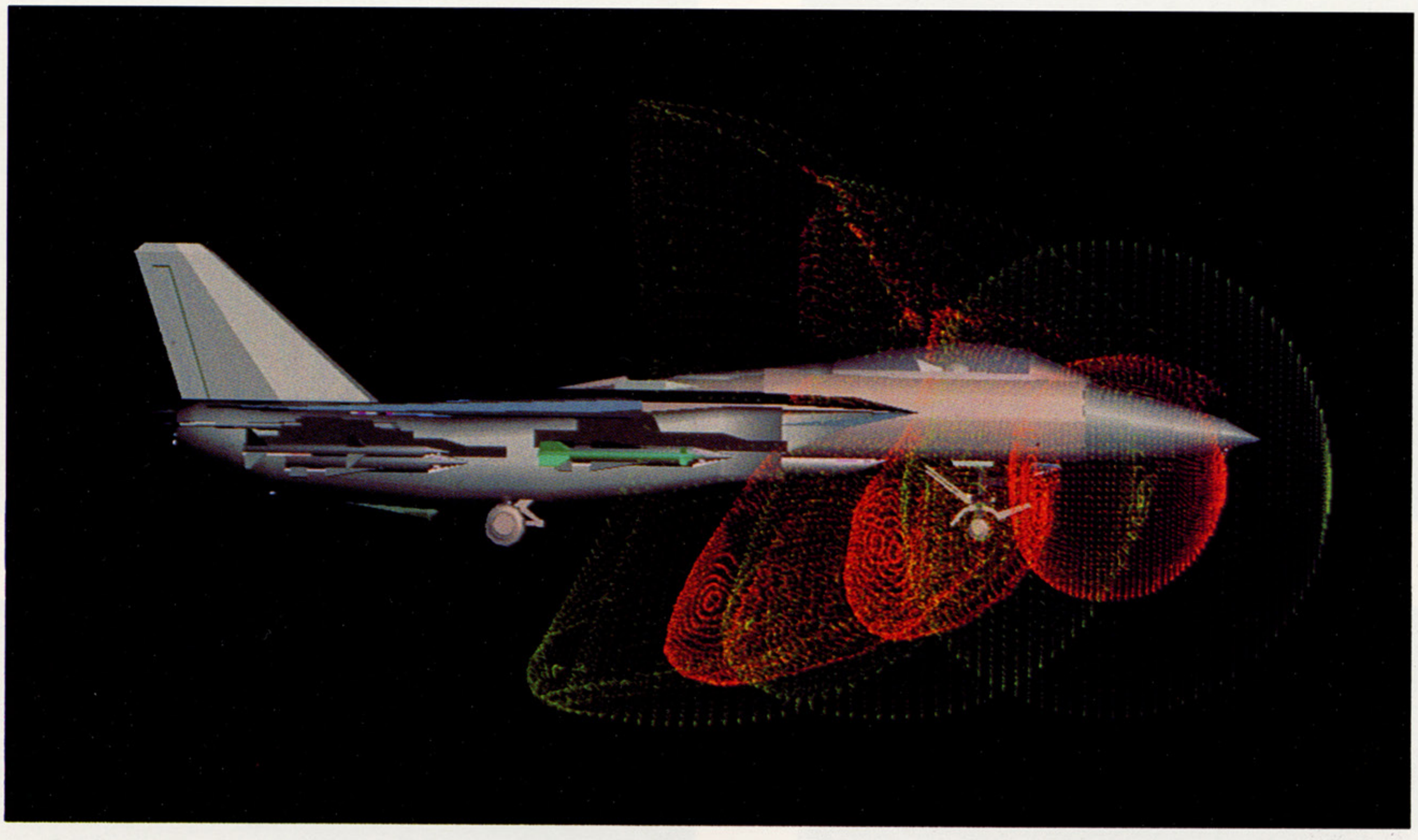


Figure 7: The flow over an F-14 was defined by a coarse grid of one million cells. Spherically distributed particles were deposited near the nose of the airplane and tracked as a function of time. Later, new particles were introduced. 120 images were generated and compressed and then transported (ftp) to a Dell 386 PC with a VGA board and a 9 megabyte RAM disk). This animation sequence was then displayed at approximately 16 frames per second with an 8-bit color quality close to the original 24 bit image shown here in Figure 7.

using a palette generation technique developed at the Weapon's Laboratory (Gleisher/Conley), a 24 bit image could be displayed as an 8 bit image (not-dithered) and in most cases appear as good as the original 24 bit image.

As an example, 120 frames of 1280 x 1024 x 24 bits was transferred from the Cray-2 to a Dell 386 personal computer with a VGA board and a 9 megabyte RAM disk. Transfer time was approximately ten minutes. The compressed file on the Dell required from 3 to 5 megabytes. Decompressing a reduced image (640 x 480 or 320 x 240), the results were displayed on the 386 at approximately 6 to 16 frames per second. Using a Silicon Graphics or Sun machine, one could open a window to communicate with another host computer, remotely executing the simulation program, and, in a second window, simultaneously download the image files and display the results.

Employing these same compression techniques to the flow solution (density, velocity components, energy), it was found that for NEWTUN compression ratios were in the 90 percentile. For simulations employing body-conforming grids, compression ratios were only 25 percent to 50 percent with a 1 percent loss in accuracy.

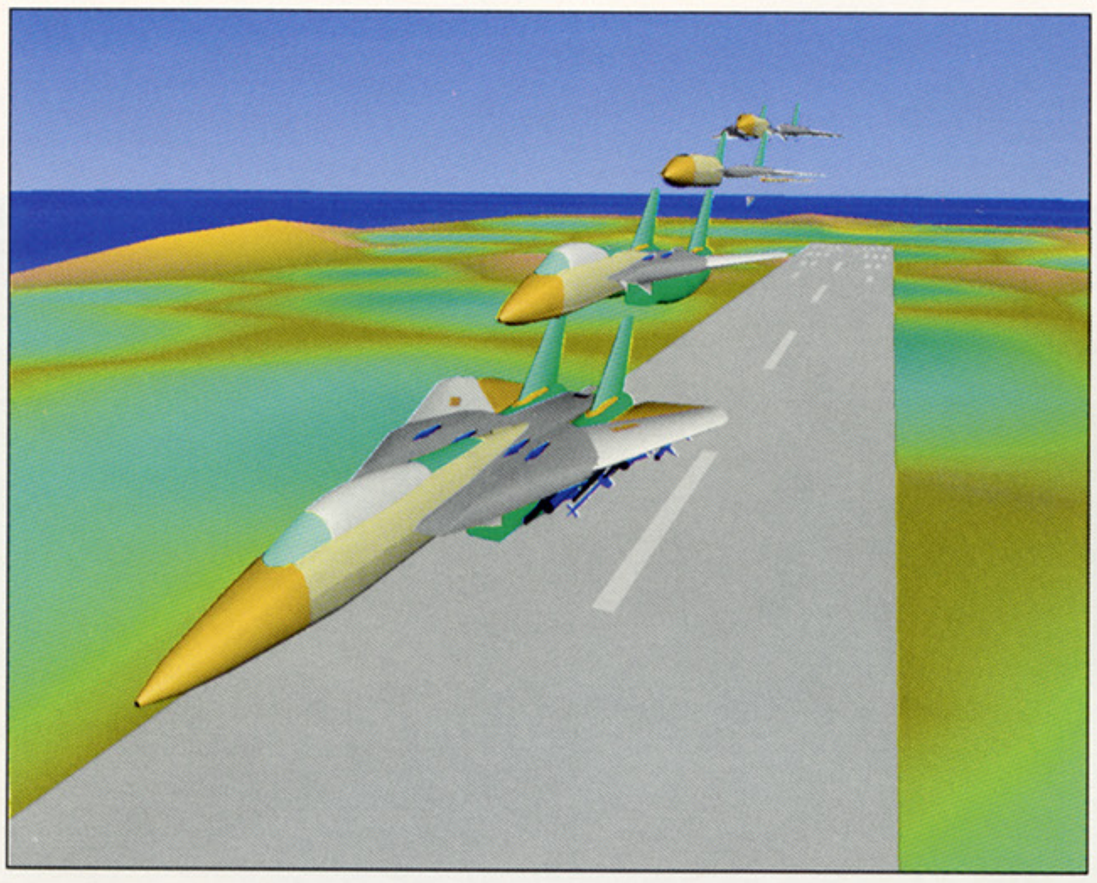


Figure 8: For faster drawing speeds, REPLIMOVIE provides for object Declutter Control, whereby the further away an object the less detail the Graphics Device processes. The Personal IRIS used to generate all REPLIMOVIE images had neither turbo nor HW floating point.

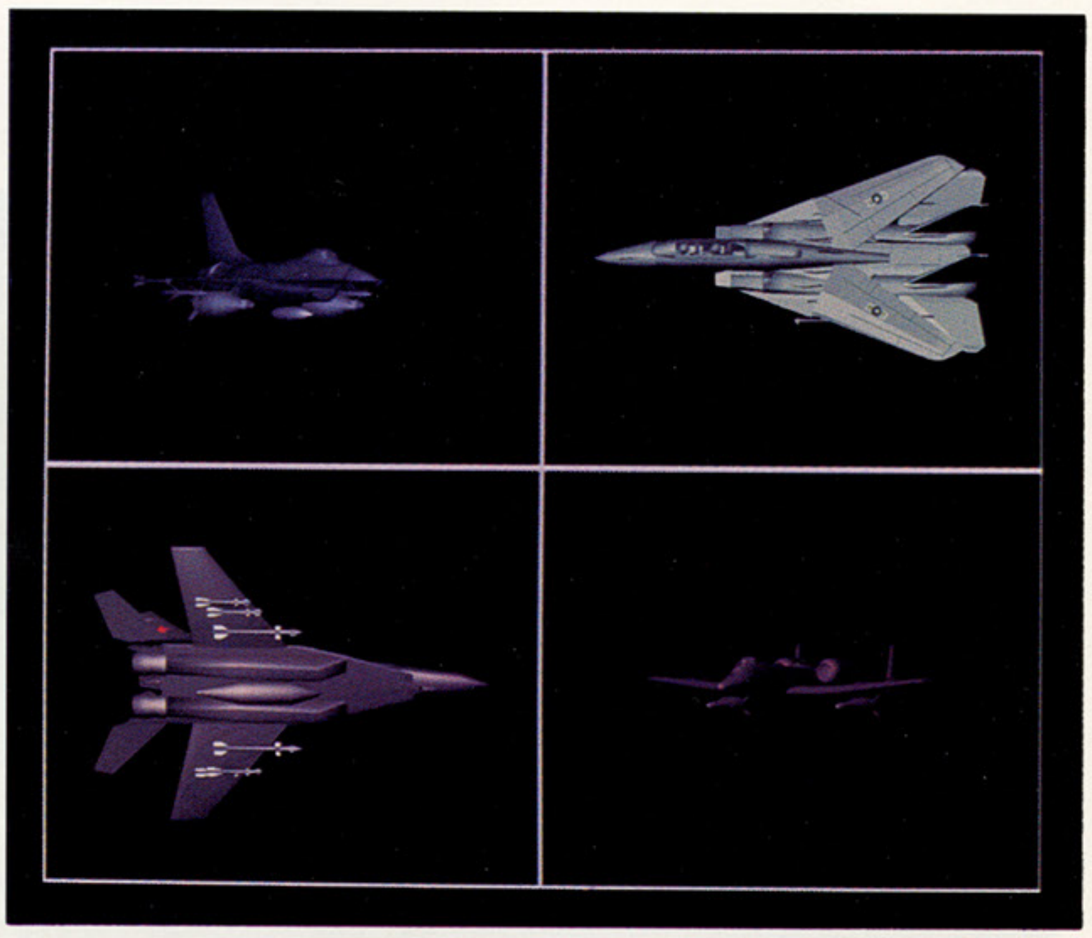
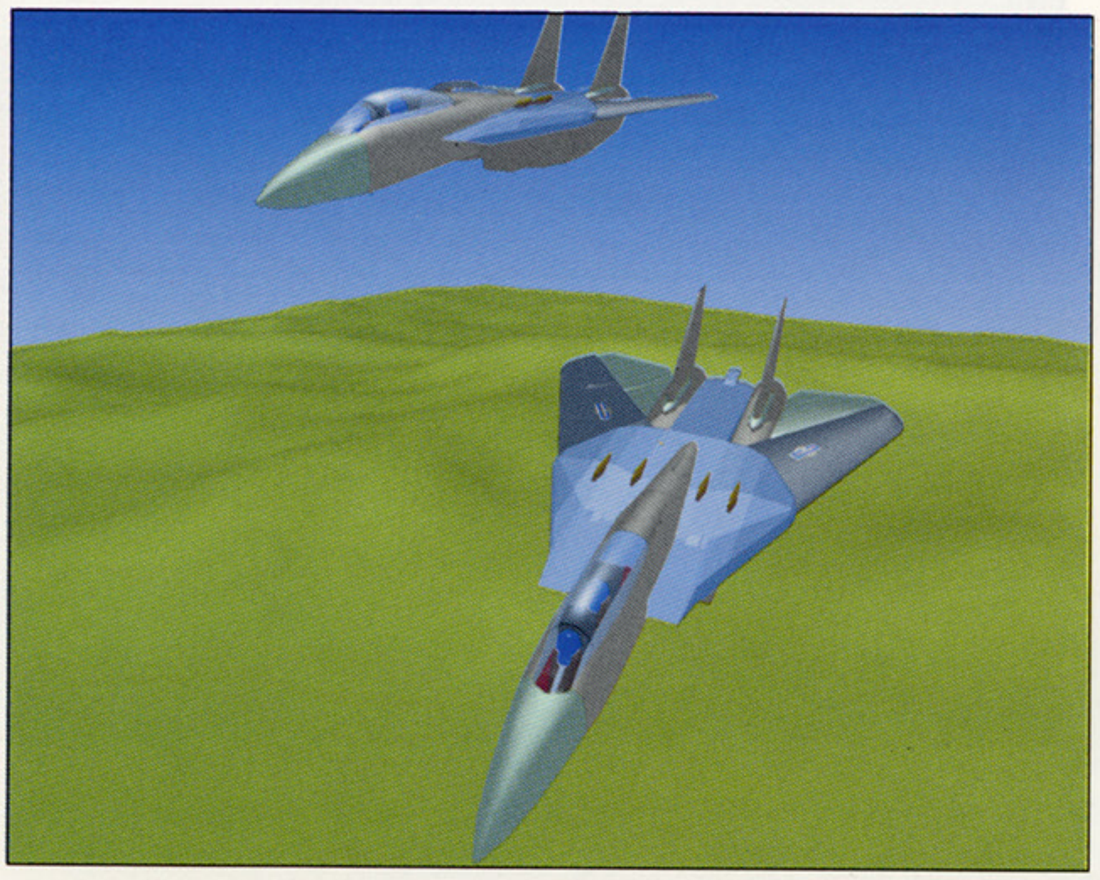


Figure 10: Four airplanes are shown: F-16 (top-left), F-14 (top-right), MIG-29 (bottom-left) and A-10 (bottom-right). The planes were generated using REP-TILE on a CRAY-2 in 1/10 second. REP_TILE runs on other computers including the IRIS-4D, Convex, and the SUN-3 and 4. REP_TILE employs a multiple light source algorithm, producing 24-bit rgb images. Bodies can be produced by MOVIE-BYU, WAVEFRONT, AUTOCAD, and similar programs. The animation scripts are supplied from a user input file.

Figure 9: Displayed scenes can contain great quantities of graphic data and yet be rendered quickly by using new multi-processor hardware and REPLIMOVIE's Virtual Universe™ feature.

Figure 11: Simulated motion data is visually animated for aircraft maneuvers, vehicle dynamics, and robotics using REPLIMOVIE's Voyager™ option. REPLIMOVIE was authored by James A. Squires.



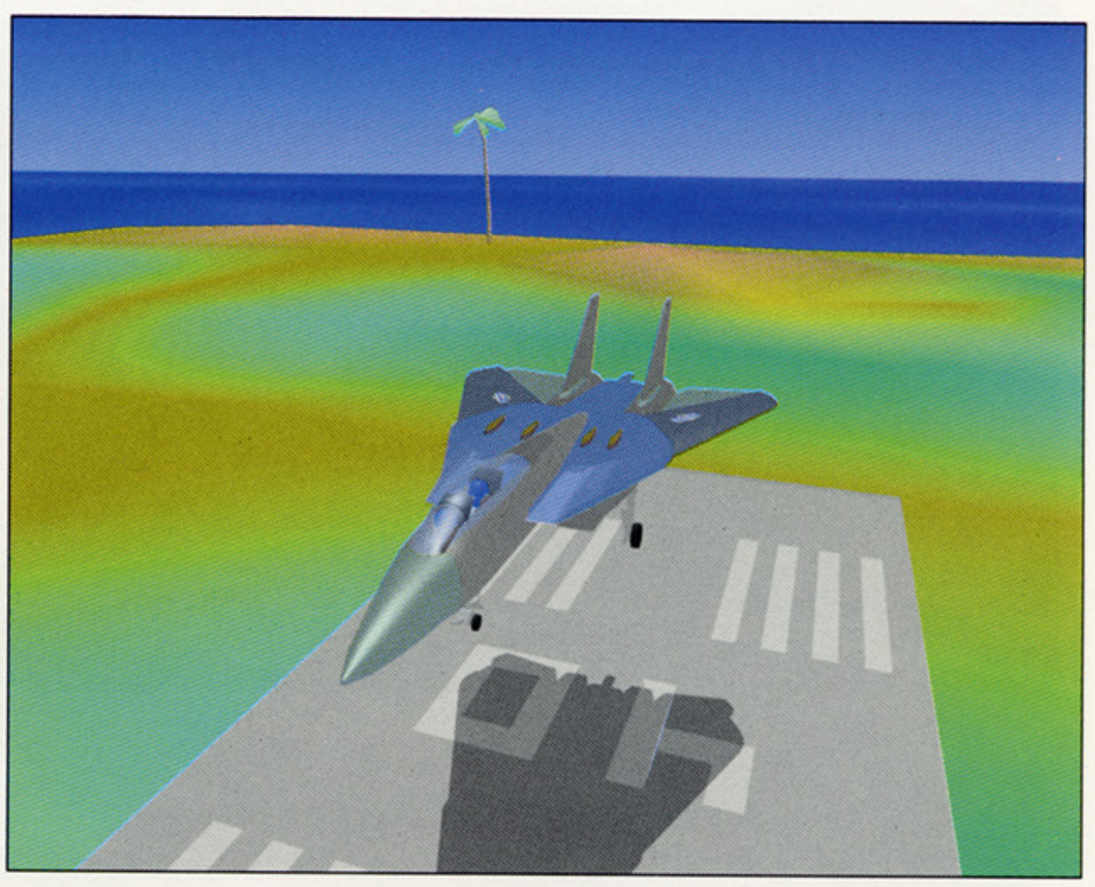


Figure 12: For high-quality rendering, features such as shadows, texture mapping, reflections, Phong shading, and anti-aliasing for 24 bit and 8 bit dithered images can be achieved using REPLISHADE. The F-14 pictured in the REPLIMOVIE and REPLISHADE images contains approximately 2000 4-point polygons.
Image courtesy of James A. Squires

Issue No. 6: What Other Forms of Simulation are Practical?

If one defines computer graphics simulation as any form of data displayed in a dynamic manner, the potential scope of this capability quickly becomes apparent. The range of uses for computer graphic simulation is growing as the hardware and software technology advance. Flight maneuvers can be "played back" for both instruction and study. The data from aircraft "black boxes" can be used to vividly recreate airplane disasters for use in preventative failure analysis. For the same purpose, automobile accidents can be recreated and mechanical or road conditions analyzed. Assembly line robots can be simulated for interactive programming. The effect of earthquakes on buildings or elevated highways can be depicted in order to identify stress points while the structure is still in the design phase, thereby saving hundreds if not thousands of lives. On a smaller scale, all manner of mechanical parts can be analyzed and redesigned to optimize their strength, efficiency, and economy.

A program such as REPLIMOVIE, with the aid of an intuitive definition and simulation command language, allows the user to interactively describe the graphic environment and motion criteria for innumerable applications which will benefit from graphic simulation.

Issue No. 7: Graphic Realism: Do I Really Have the Time?

Shaded images were produced employing four techniques: (1) Z-buffer, (2) Z-buffer in hardware (SGI library), (3) scan-line renderer and (4) ray-tracing. Techniques (1), (2), (3) and (4) were written in FORTRAN, machine code, C and PASCAL respectively. Figures 10, 11, 12, and 13 represent images produced by these techniques. Airplanes consisting of approximately 3400 polygons were used. Employing a z-buffer on a Cray-YMP, using a multi-tasked, vectorized version of REP_TILE, each airplane required approximately 1/15 of a second to compute and display. Running REP _TILE on a Personal IRIS (not using the IRIS graphic engine) required about two minutes per plane. Executing "Replimovie" which employs the IRIS graphic engine and light model, each airplane of approximately 2100 polygons rendered within a 512 X 512 image, required .065 seconds on the Personal IRIS and 0.025 seconds on the Power Series IRIS. REPLISHADE, using the same size image on the Personal IRIS, produced each plane in 6 minutes 7 seconds, and on the Power Series in 2 minutes and 29 seconds. However, REPLISHADE calculated shadows, Phong shading, and anti-aliasing which is not offered by the first two techniques. A ray-traced image developed for the Cray series

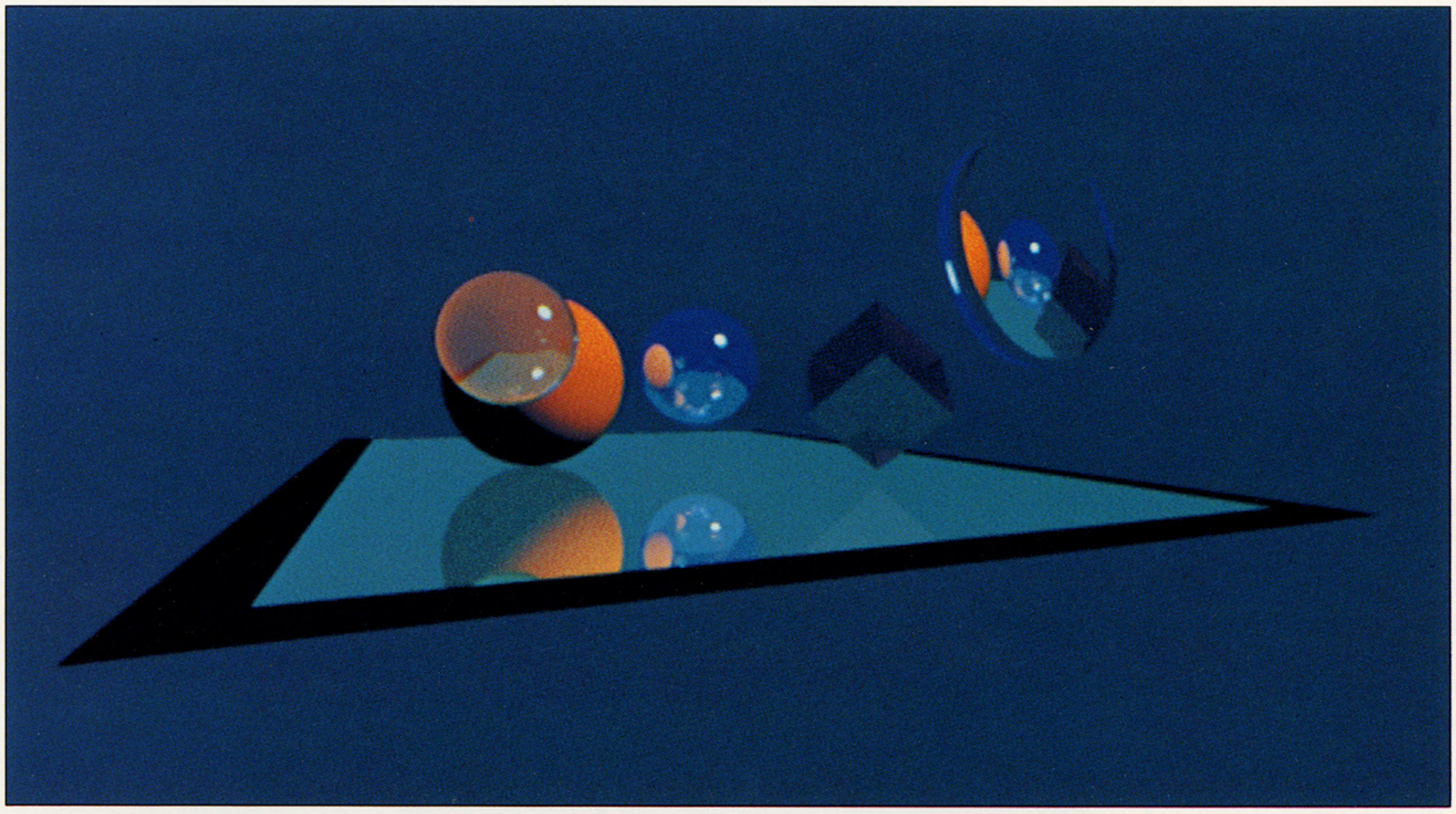


Figure 13: Ray traced images of reflecting and refracting objects, showing the level of realism possible by correctly rendering transparent surface boundaries and internal propagation. The program calculates transparency and translucency properties based on coefficients representing the material and on the distance of material traversed. Multiple reflections and refractions are retained until a terminating condition is encountered; ray strikes an opaque object, ray strikes a light source, ray has intersected 50 surfaces, or accumulated attenuation of light reaches 0.001. Image resolution of 1280 X 1024, with recursive descent for up to 4 times oversampling for anti-aliasing, required 2 to 3 minutes per image on a Cray-2. The program was authored by Robert W. Conley, Jr. computers will also be used to render airplane and flow field data. All of the above techniques employed multiple light sources and produced 24 bit r,g,b images.

In conclusion, it was found that visual simulation systems are optimized when personal preference is combined with a computer environment appropriate to the task at hand. Frame buffers and graphic workstations serve complementary roles. Compression of image data, multi-processing, and vectorization significantly improve performance. Application Specific Integrated Circuits (e.g the IRIS graphics engine) can achieve performance levels *competitive* with high level graphics programs running on a Cray YMP. Most importantly, graphic realism can be accomplished in stages and need not be time consuming or costly.

For further information on the programs or the subject matter discussed in this article please contact: Creative Visual Software, Inc., Post Office Box 20638, San Jose, California 95160 (408) 997-1621, FAX: (408) 268-0766. ■

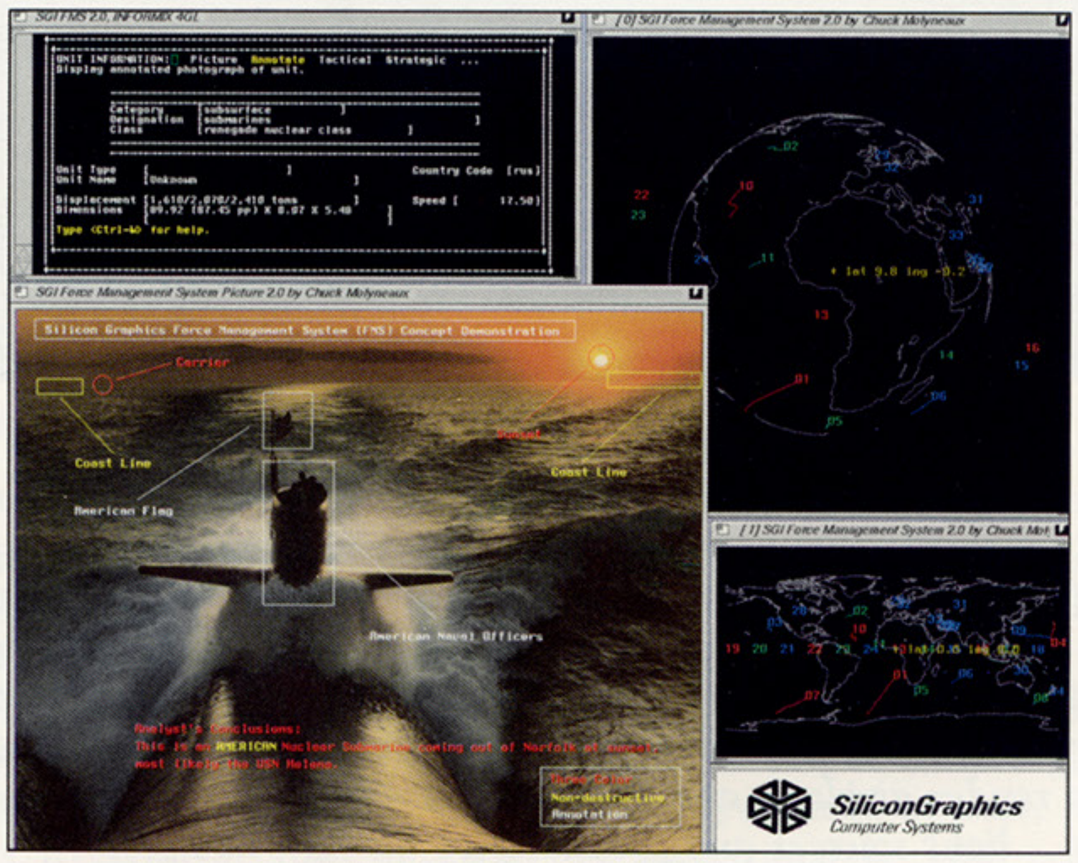
Laurence Feldman holds a Ph.D. in Aerospace Engineering from Auburn University, Alabama and is the author of REPLICORE, NEWTUN, REP_TILE, SHO_FLO and POST_FLO.

feature

DATABASE VISUALIZATION

BY CHUCK MOLYNEAUX

People need information to make decisions; organizing the information is the key to using it successfully. Database companies have capitalized on this need. At the same time the emergence of three-dimensional computing has brought with it special requirements. The ability to capture video signals and mix live color images further complicates the process of data gathering and display. However, combining all of these capabilities holds the promise of allowing users to interact with convincing simulations of the "real world."



This concept is still in its infancy, but both database technologies and visualization platforms are at the point where such a blending of information is possible. Silicon Graphics together with Informix, now provides a solution by offering *Visual Database Lab (VDL)*, a product that lets application developers blend database information with images and sound in an interactive, multiwindowed, multiprocessing display environment.

VDL runs on all Silicon Graphics 4D workstations and servers, and is supplied to users in source-code form. It is written in a combination of C code and Informix 4GL, and requires the Informix relational database.

Silicon Graphics supplies a prototype application called the Force Management System (FMS) 2.0 with *VDL*. The program allows users to access a database of military deployment information by clicking on geographic areas on a three-dimensional world map. The FMS uses standard Informix database features and program techniques, and represents real-time 2D and 3D world views in a package that simulates present day Department of Defense activities. The FMS demonstration database contains no actual military data.

In the illustration an IRIS screen displays the multiple features of FMS. The upper right side contains a 3D globe and the lower right side shows a 2D map. Both contain identical tracking and location information (i.e. ships, planes, satellites, armored vehicles, terrain and facilities) with the 3D globe adding altitude information to the equation. The upper left contains an Informix 4GL window which displays textual information as a result of a pick generated query into one of the potential databases. The bottom left contains a fully commented photograph of the track selected. In this example it is a submarine.

Users can create an application from scratch with *VDL*, but it's much easier to take the FMS demonstration database and modify it to meet a particular need. Likely commercial applications of the *VDL* package include the development of 3D molecular modeling with movement simulation, insurance claim databases with photo images and geographic locators, CAD/CAM applications, and fleet vehicle tracking and maintenance.

VDL initially will be delivered with extensive development documentation and the FMS prototype application containing the *VDL* source code. Priced at under $10,000, *Visual Database Lab* was released in January of this year. ■

Chuck Molyneaux is a systems engineer at Silicon Graphics' Bethesda, Maryland office.

CD -ROM Available Now !

■ ISO 9660 ■ High Sierra Format ■ DMA Terrain Data

The Introl STERLING 600 CD brings the world of CD-ROM to Silicon Graphics workstations, including Defense Mapping Agency (DMA) digital terrain data.

Hundreds of CD-ROM applications

Other compatible CD-ROM's include medical data bases, clip art files, Jeppeson air charts, U.S. Geological Survey data, Cambridge Scientific medical databases, government data, encyclopedias, and even your favorite audio CD.

The STERLING 600 CD includes Introl software to use the existing SCSI port on any SGI workstation. Auto install software makes installation a snap. Standard UNIX/ SGI commands can then be used to access the CD-ROM.

The STERLING 600 CD is available in a matching SGI drive tower enclosure or in a desktop case for Personal IRIS systems.

Introl's Family of SGI Peripherals

Available for immediate delivery:

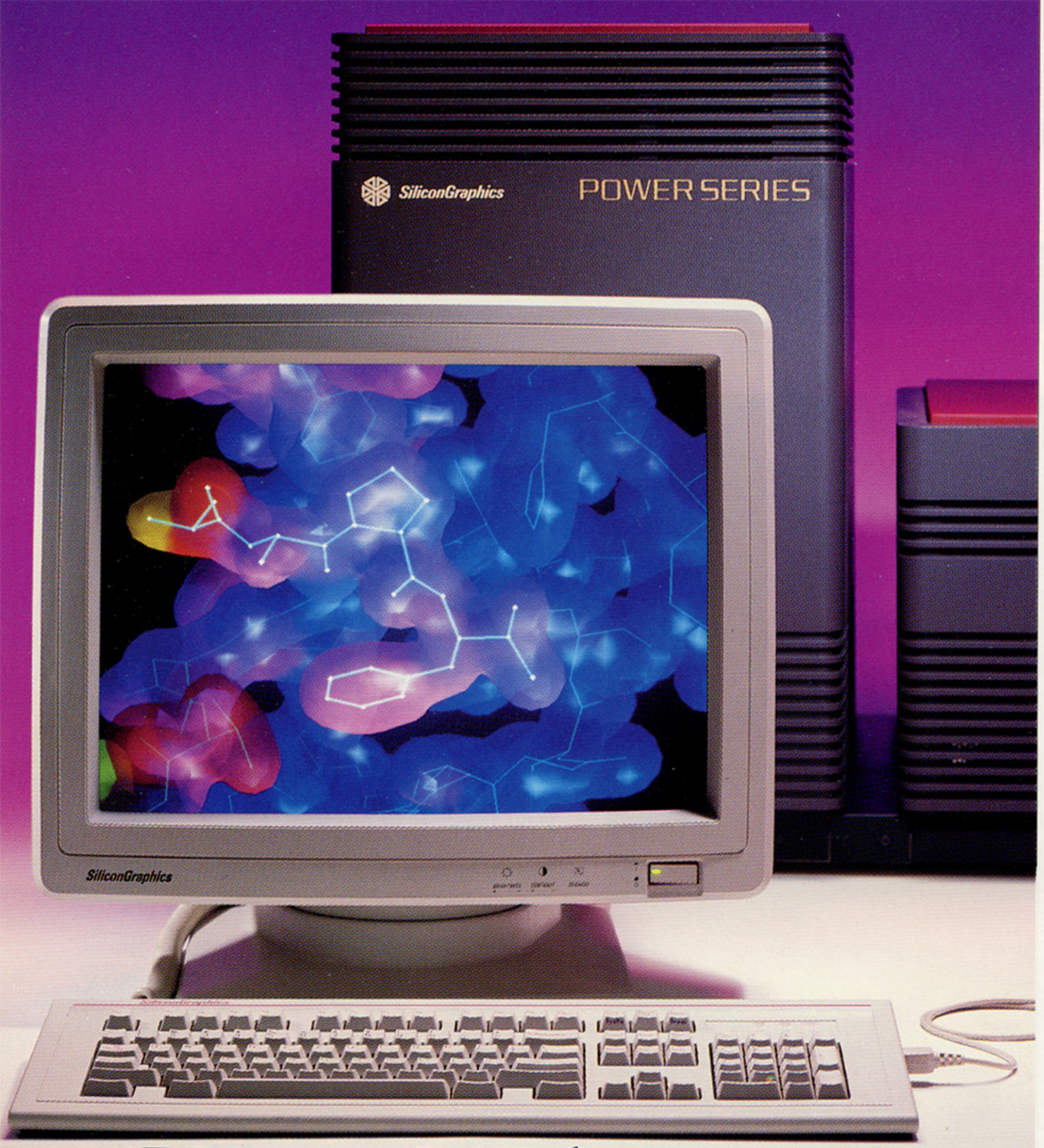
- 650 MB Erasable Optical Subsystems
- 2.3 GB 8mm Tape Subsystems
- 1.2 GB DAT Tape Subsystems
- Removable 396 MB to 3.4 GB Winchester Disk Subsystems
- Custom Subsystem Designs

Experience the Introl difference: Sterling quality and performance backed by people who care.

Call Us For A Quote On Your Custom Peripheral Needs - (612) 631-7600

2675 Patton Road
St. Paul, MN 55113
(612) 631-7600 FAX (612) 631-7802

Introl Corporation



How 4 Of The Top 5 Chemical Companies See Computational Chemistry.

The fact is, there are more companies solving chemistry problems with Silicon Graphics® workstations than all other workstations combined.

It's easy to see why: Silicon Graphics workstations and servers are the industry standard for 3D computing. They run more major chemistry software applications than any other workstation. And, they run them far faster.

See for yourself why Silicon Graphics is *the* choice for computational chemistry.

Get the data.
A quick, toll-free call gets you a thorough data package with benchmark results, our customer list, available chemistry software, technical literature, and more.

Data Package HOTLINE:
1-800-952-6300 S200
In Canada, call 416-674-5300

SiliconGraphics®
Computer Systems

A few of the organizations that have made Silicon Graphics the leading workstation for chemistry:

- DuPont
- Dow Chemical
- ARCO
- Monsanto
- BASF
- Amoco
- Allied Signal
- Kodak
- Chevron
- American Cyanamid
- Abbott Labs
- Agouron Pharmaceuticals
- Beechams Pharmaceutical, U.K.
- BioCryst, Ltd., Alabama
- BioDesign
- Biospan
- BIOSYM Technologies, Inc.
- Bolt Beranek and Newmann
- Burroughs Wellcome
- CELLTECH, U.K.
- Chiron
- CNRS GIF, France
- Eli Lilly
- E.R. Squibb
- Farmitalia
- Genentech
- Henkel, W. Germany
- Homburg DeGussa Pharma Group, W. Germany
- Hybritech
- IBM
- ICI, Delaware
- Janssen Pharmaceutica, Belgium
- John Von Neumann National Supercomputer Center
- K.U.L. Esat, Belgium
- Kraft, Inc.
- Merck
- Molecular Structure Corporation
- N.I.E.H.S.
- N.I.H.
- Nabisco Brands, Inc.
- NOVA, Baltimore
- Novo Industries
- Pfizer Ltd., U.K.
- Polygen Corporation
- Procter and Gamble
- Protein Design Labs
- Sandoz
- Shell
- Smith Kline & French
- SRI International
- Stuart Pharmaceuticals, (ICI) U.S.
- Sumitronics, Inc., Japan
- Syntex
- Taisho Pharmaceuticals, Japan
- TCG Systems Automation PTY, LTD., Australia
- Tripos Associates
- Upjohn Company
- Zymogenetics

We run the industry's broadest selection of 3D software for chemical research, including:

AMBER	MADNMR
BIOGRAF	MIDAS
Bio-Explore	MMS
Bio-Gromos	MM2
CHARMm™	MOLCAD
CHIRON	MOPAC
Discover	NMR1
DSPACE	NMR2
FRODO	POLYGRAPH™
FTNMR	QUANTA™
GAUSSIAN 85	RIBBONS
GEMM	SPARTAN
GROMOS	SYBYL®
Insight	TOM
MACROMODEL	X-PLOR

Silicon Graphics Marketing Services
707 California Street
Mountain View, CA 94041

Screen image: Alpha Lytic Protease active site. Data courtesy of David Agard and Rodger Bone. Illustration by Julie Newdoll and Greg Couch, UCSF CGL.

feature

THE HIDDEN CHARMS OF Z-BUFFER

Effective use of IRIS Z-buffer hardware requires familiarity with techniques for hidden surface removal. Several little-known tricks can also be useful. Advice on both scores follows.

BY KURT AKELEY

IRIS graphics systems utilize a hardware Z-buffer to render images with hidden surfaces eliminated. The Z-buffer hardware is configurable, meaning not only that its use can be optimized for hidden surface elimination, but that additional operations are also possible. This article describes the Z-buffer, suggests how it can be used for simple hidden surface elimination, and provides solutions for some of the more complex operations, including applying decals, highlighting surface tessellations, and generating line drawings with hidden lines removed. Many of the techniques described here are applicable to all IRIS graphics systems. The hardware-specific approaches, all of which have been flagged as such, work with the GT/GTX architecture.

The Z-buffer is a set of 24-bit numbers, each of which relates to a pixel on the screen. Use of the Z-buffer is enabled with the **Zbuffer** command:

zbuffer(TRUE);

Points, lines, polygons, and characters are reduced during rendering to pixels, each with its own color and x and y coordinates. Also, when the Z-buffer is enabled, a z coordinate is computed for each pixel. This coordinate effectively specifies the distance from pixel to eye.

Before a pixel's color is written to the framebuffer, its z value is compared with the value already stored in the Z-buffer. If the incoming pixel is nearer (that is, if it has a lower z value), its color is written into the framebuffer and its z is written into the Z-buffer. If it is farther (that is, if it has a greater z value), the incoming color and z are discarded. Thus, at any point in the drawing, the values in the Z-buffer represent the distance to the item which currently is closest to the eye. By first clearing the Z-buffer to the maximum z value and then rendering all primitives using the **zbuffer** algorithm, an image including only the nearest surfaces to the eye is produced.

Mapping Transformed Z to Screen Z

Mathematically, z coordinates are treated just like x and y coordinates. After transformation, clipping, and perspective division, they occupy the range -1.0 through 1.0. Just as you specify a viewport transformation to map x and y from this range to a window's boundaries, so must you specify a mapping of z coordinates. Because the 24-bit Z-buffer on the GT and GTX graphics systems is unsigned, it supports the range 0x000000 through 0xffffff. Only values in the range 0x000000 through 0x7fffff are correctly iterated during rendering, however, so only 23 of the 24 Z-buffer bits typically are used. (We'll demonstrate a use for the 24th bit later in this article.)

By default, GT and GTX graphics systems map those pixels nearest the eye (*i.e.* the ones at the near clip plane) to 0x000000, while the pixels farthest from the eye (*i.e.* , those

at the far clip plane) are mapped to 0x7fffff. This can be changed with the **lsetdepth** command:

lsetdepth(0x000000,0x3fffff);

Called like this, the mapping will be reduced to use only half of the hardware range.

Recall also that each Z-buffer location must be initialized to *farthest* prior to rendering. Application performance is increased if the Z-buffer and the framebuffer are initialized at the same time. To accomplish this, replace the code sequence:

cpack(0xffffff);
clear();
zclear(); /*clears Z-buffer to 0xffffff*/

with:

czclear(0xffffff,0xffffff);

On GT and GTX hardware, **czclear** operates on the framebuffer and the Z-buffer simultaneously only if the color argument (first) and the depth argument (second) are the same (see the **czclear** man page for details). In this example, we've chosen a white background, and have cleared the Z-buffer to its maximum value. The fact that this value is greater than the greatest rendered value (0x7fffff) has no effect on the rendering operation.

It is also possible to achieve simultaneous clears with a black background, like so:

czclear(0x000000,0x000000);

For this to provide correct results, however, both the z mapping and the z comparison must be reversed:

lsetdepth(0x7fffff,0x000000);
zfunction(ZF_GEQUAL);

This mapping gives nearer pixels greater Z-buffer values than farther pixels, with pixels at the near plane mapped to 0x7fffff, and those at the far plane mapped to 0x000000 (the value the Z-buffer was cleared to). In order to correctly select the nearest pixel, the comparison function is simply reversed to select the pixel with the greater z value. (The default **zfunction** is ZF_LEQUAL.)

We previously asserted that a Z-buffer value, in effect, specifies the distance from pixel to eye. While this is true, the relationship between distance and z is linear only in an orthographic projection. In the case of a true perspective projection, such as:

perspective(*fovy*,*aspect*,*near*,*far*);

the relationship is non-linear, sometimes very much so.

The degree of non-linearity is controlled by the ratio of *far* to *near* in the **perspective** call. The greater the ratio, the greater the non-linearity. Figure 1 plots Z-buffer value as a function of eye-to-pixel distance for several ratios of *far* to *near*.

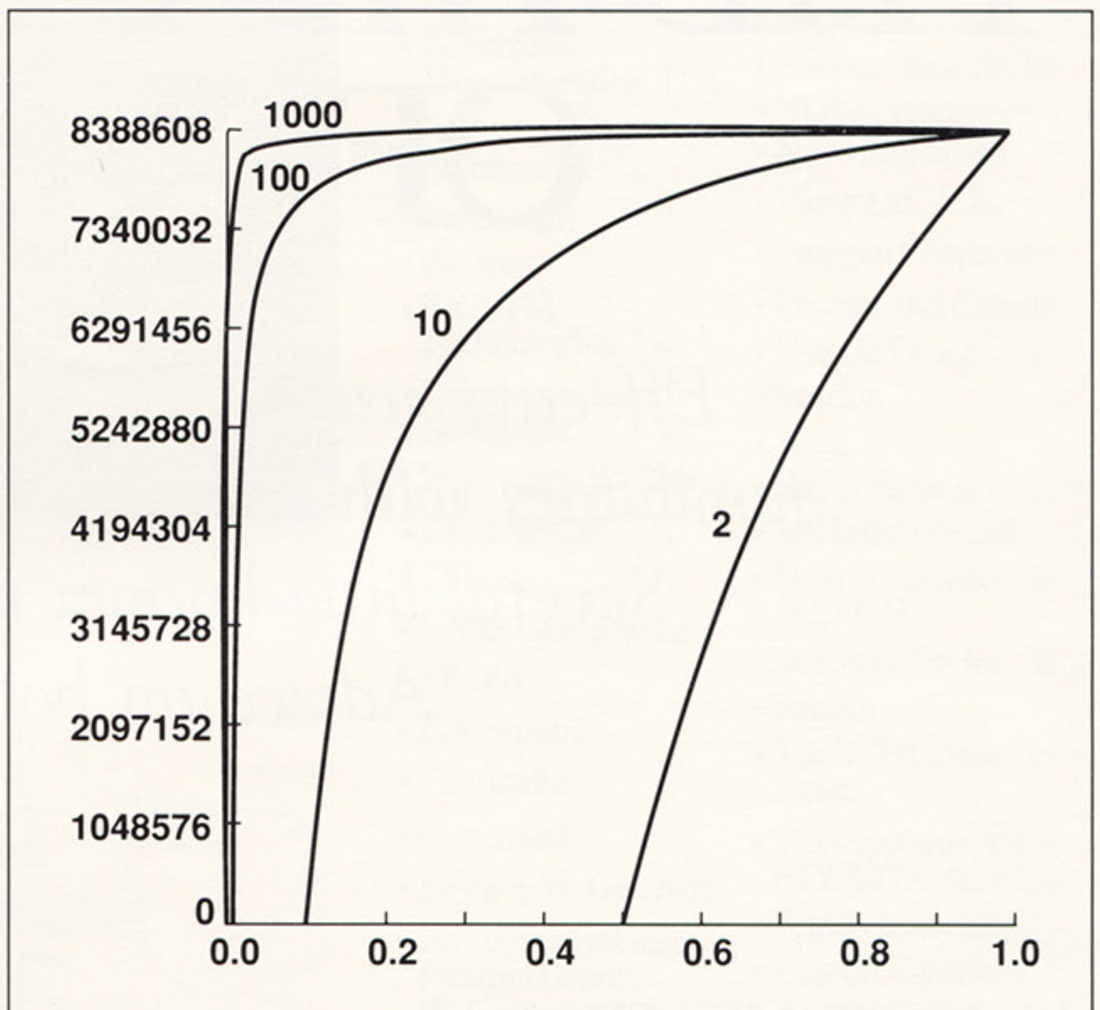


Figure 1: The relationship of screen z to eye z, as a function of the ratio far/near in the perspective call. Plots are for ratios 2, 10, 100, and 1000. Notice that, as the ratio approaches 1000, almost all the Z-buffer resolution (vertical axis) is concentrated near the viewer (horizontal axis).

In practice, non-linearity increases Z-buffer precision in a small range adjacent to the near clipping plane, and reduces precision throughout the rest of the viewing volume. While some precision increase near the viewer can be desirable, the effect of substantial non-linearity is to defeat Z-buffer operation throughout much of the viewing volume. Our experience is that ratios greater than 1000 have this undesired result.

The ratio of *far* to *near* is most easily controlled simply by moving the near clipping plane *away* from the eye position. Note that changing *near* in the **perspective** call has no effect on the projection of x and y, and so has little effect on the resulting image. Instead, its effect is limited to the position of the near clipping plane and the projection of z values. The moral? Always move the near plane as far from the eye as possible.

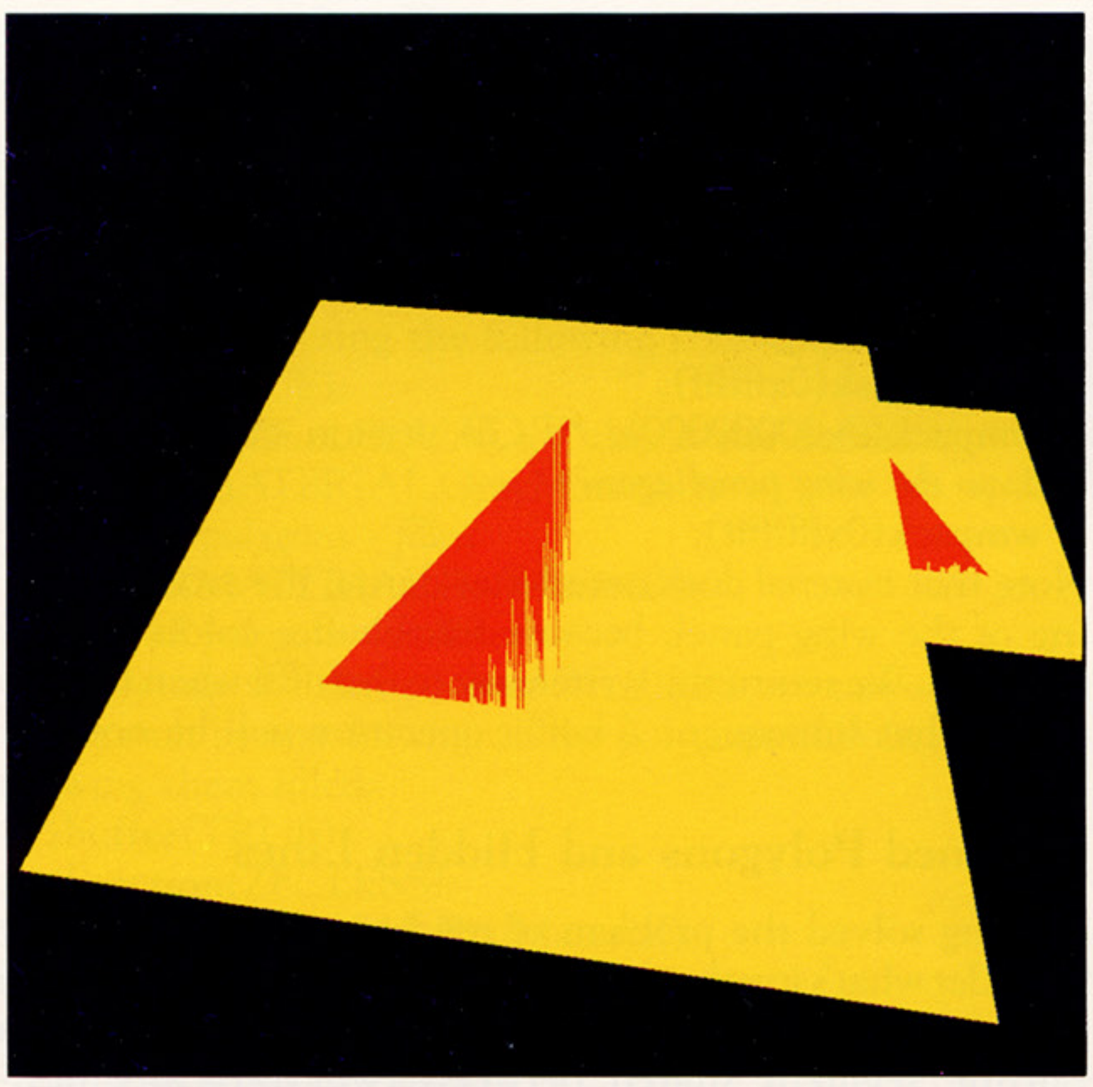


Figure 2: Triangular decals applied to quadrilaterals using the standard Z-buffer algorithm. Errors in z interpolation result in tearing.

Figure 3: The same decals as figure 2, but applied using the decal algorithm. The tearing is eliminated.

Z-Buffered Decals

Because of the limitations of Z-buffer precision, and because the precision that exists is *lost* due to non-linear mapping, the IRIS Z-buffer has difficulty in computing the correct intersection of nearly coplanar objects. As a result, objects intended to *be* coplanar, such as a stripe on a runway, or a marking on an airplane wing, are rendered very poorly when nothing more than the standard Z-buffer algorithm is used (*i.e.* the runway shows through the stripe, and the wing shows through its marking). We refer to this problem as the *decal* problem.

GL commands **writemask** and **zwritemask** support a simple solution to the decal problem. In effect, they allow the use of a *painter's algorithm* (the last object written wins) among coplanar polygons, and yet they use the Z-buffer algorithm to integrate coplanar polygons into the scene. As an example, let's consider a red triangle drawn on a gray quadrilateral piece of an airplane's wing. Assuming the Z-buffer has been correctly initialized and that normal Z-buffer rendering is progressing, let's now add the wing quadrilateral, together with the triangular marking, to the scene:

1) Draw the gray quadrilateral using the normal Z-buffer comparison, but *don't change any values in the Z-buffer.*
2) Draw the red triangle, again using the normal Z-buffer comparison. Again leave the Z-buffer itself unmodified.
3) Finally, draw the gray quadrilateral a second time, using the normal Z-buffer comparison. This time, update Z-buffer values, *but make no change to the framebuffer colors*. The red triangle will remain accurately rendered within the boundaries of the gray quadrilateral.

Having rendered the wing piece, the remainder of the scene can be completed with the help of the standard Z-buffer algorithm. Figures 2 and 3 illustrate the difference between the results obtained using the standard Z-buffer algorithm (Figure 2) and those achieved using the decal algorithm (Figure 3).

The following pseudo-code sequence implements the algorithm (the bold GL commands designate real code):

zwritemask(0x000000);
lmbind(MATERIAL,GrayMaterial);
draw the wing panel
lmbind(MATERIAL,RedMaterial);
draw the triangle
zwritemask(0xffffff);
wmpack(0x00000000); /*RGB version of w**ritemask***/
draw the wing panel again
wmpack(0xffffffff);

Note that *material* does not matter during the second drawing of the wing panel, because framebuffer colors are not modified. We return the **writemask** to 0xffffffff when done to ensure that subsequent Z-buffer operation will be normal.

Outlined Polygons and Hidden Lines

Having solved the problem of rendering decals, let's now consider what's involved in *outlining* a polygon. The problem seems similar, except in this instance we must account for an outline segment shared between two (typically non-coplanar) polygons. The decal algorithm fails when a single decal is shared among multiple, non-coplanar polygons. (Can you see why?)

To solve the decal problem, we worked around the inaccuracies of z projection and iteration. Let's imagine, however, that the Z-buffer worked perfectly, meaning that coplanar objects matched z values pixel for pixel during rendering. In such a case, the *painting* effect could be had simply by selecting the appropriate **zfunction,** thus avoiding the need for **writemask** tricks. For example, selecting the **zfunction** ZF_LEQUAL (pass if less than or equal) causes the second coplanar polygon to completely overwrite the first, because the pixel z's will be equal, and equality passes the z test. Likewise, if **zfunction** ZF_LESS is chosen, the second coplanar polygon will be completely ignored, because the equal z values never will pass the test.

This demonstrates that it is possible to accurately composite coplanar objects if their pixels match z for z. All that remains is to postulate an outline method that ensures matched z values. The method exists, and is known as *hollow polygons*.

(*Definition:* A *hollow* polygon fills a subset of the pixels that would have been filled had the polygon been drawn normally. This subset is comprised of the pixels adjacent to the polygon perimeter. All filled pixels have color and z

values identical to their counterpart pixels in the filled polygon.)

When a filled polygon is composited with its hollow counterpart, **zfunction** can be used to accurately control the results because the *z* values of shared pixels are identical. We can, for example, draw a gray solid object with each polygon outlined in red using the following code sequence:

```
zbuffer(TRUE);
zfunction(ZF_LEQUAL); /*this is the default*/
lmbind(MATERIAL,GrayMaterial);
draw all polygons - filled
lmbind(MATERIAL,RedMaterial);
draw all polygons - hollow
```

If we change the **zfunction** to ZF_LESS, we can obtain the same result by first drawing all the polygons hollow, and then drawing them filled:

```
zbuffer(TRUE);
zfunction(ZF_LESS);
lmbind(MATERIAL,RedMaterial);
draw all polygons - hollow
lmbind(MATERIAL,GrayMaterial);
draw all polygons - filled
```

In both cases, all hidden surfaces are correctly removed and all outlines are drawn without pixel errors.

A hollow polygon primitive is available in Silicon Graphics' new *PowerVision*. However, until you have *PowerVision*, you'll have to generate the hollow polygons yourself. This is best done using the 24th bit of the Z-buffer as a *stencil* bit to control whether a pixel is writable or not. By doing this independently of the contents of the 23 depth bits, it's possible to stencil and Z-buffer simultaneously. (WARNING: This algorithm operates correctly only on GT and GTX graphics systems.)

The algorithm for hollow polygon generation is substantially more complex than any of the algorithms discussed thus far. We'll thus go through it a step at a time, first describing in common English what will happen, and then providing a pseudo code implementation. First, the Z-buffer must be set up with a reverse mapping:

```
zbuffer(TRUE);
lsetdepth(0x7fffff,0x000000);
zfunction(ZF_GEQUAL);
czclear(0x00000000,0x000000);
```

Then, to draw a single hollow polygon, first disable (for drawing purposes) all of the pixels in the polygon. Do this by setting the 24th bit of each pixel's *z* value, effectively making all values *nearer* than 0x7fffff, the mapping of the near plane:

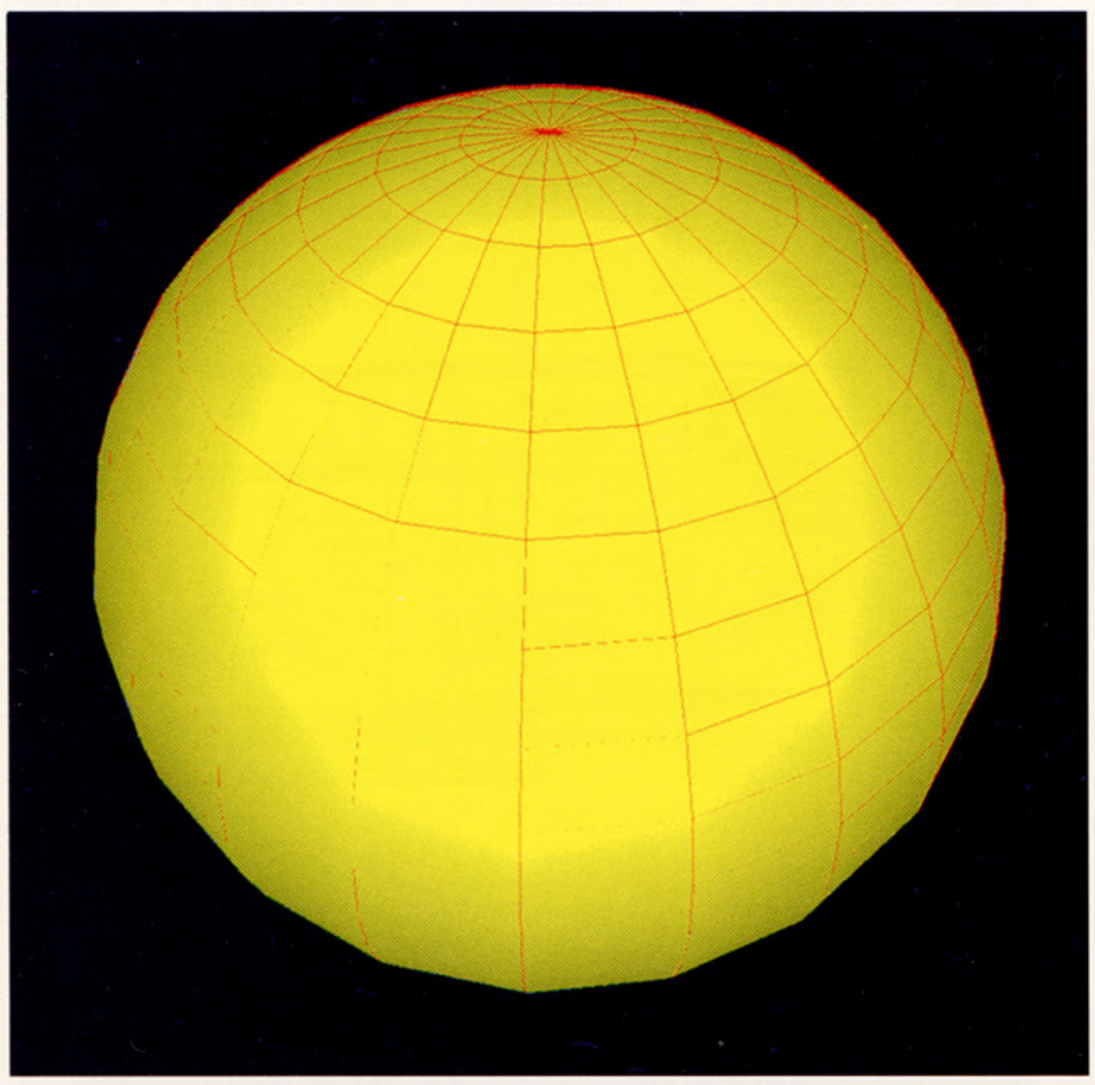


Figure 4: A solid drawn with facets outlined using the standard Z-buffer algorithm. The outlines drop out, again as a result of z interpolation errors.

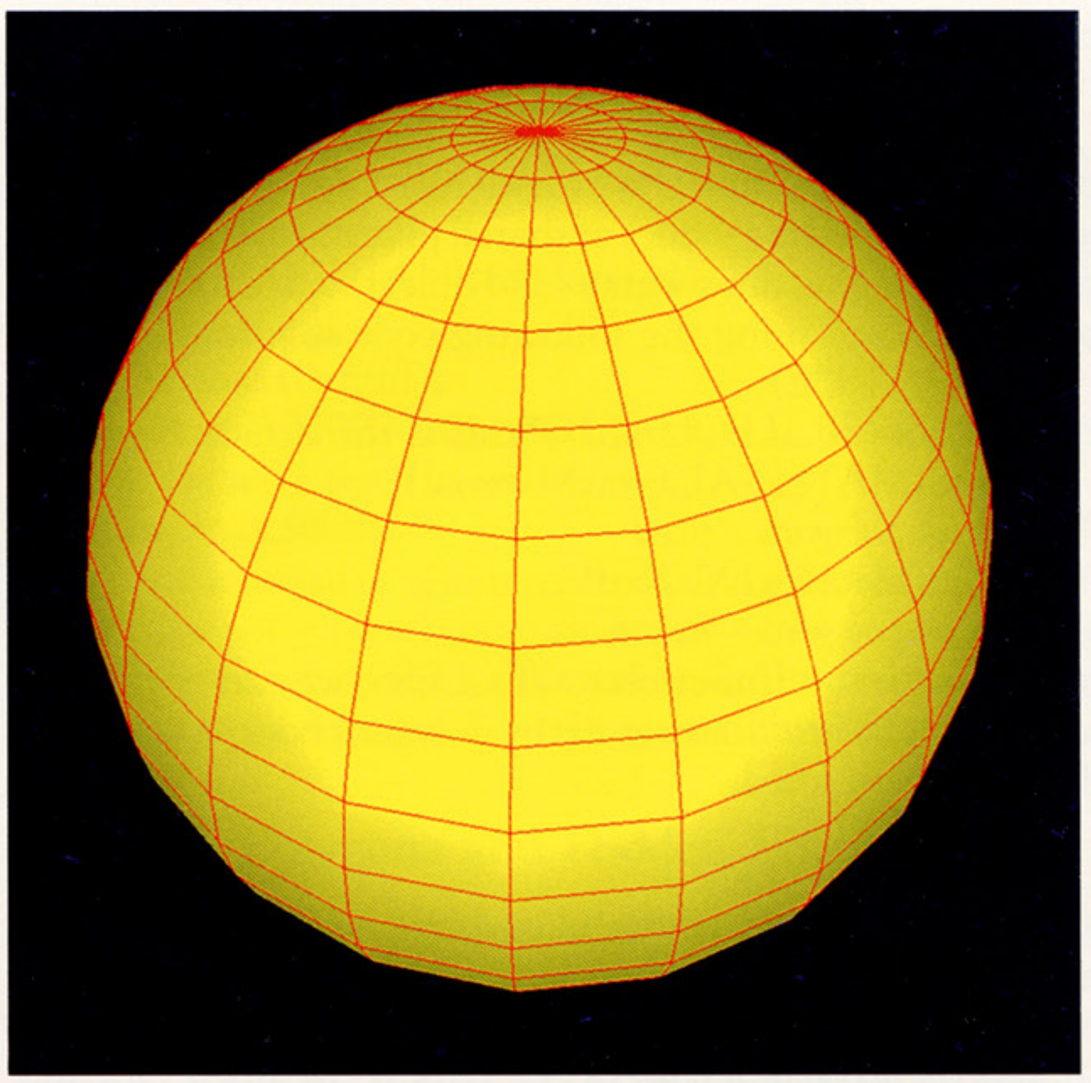


Figure 5: The same solid as in figure 5, but with outlines drawn using the hollow polygon technique. Pixel dropouts are eliminated.

zbuffer(FALSE);
backbuffer(FALSE);
zdraw(TRUE);
wmpack(0x800000);
cpack(0x800000);
fill the polygon

Now enable only those pixels on the perimeter of the polygon. Do this by drawing a width-2 line around the perimeter, clearing the 24th Z-buffer bit at each pixel drawn:

cpack(0x000000);
outline the polygon with a double-width line

At this point, pixels on the perimeter of the polygon will have their original z values. Those in the interior will have z values *nearer* than any generated during rendering, and therefore are effectively masked. Now, we simply fill the polygon using normal Z-buffer operation, which in fact fills only perimeter pixels — giving us a hollow polygon:

zbuffer(TRUE);
backbuffer(TRUE);
zdraw(FALSE);
wmpack(0xffffffff);
cpack(*color*);
fill the polygon - changing only perimeter pixel values

The 24th bit of all Z-buffer pixels that still can be set now must be cleared to ensure that future hollow polygons are not corrupted:

zbuffer(FALSE);
backbuffer(FALSE);
zdraw(TRUE);
wmpack(0x800000);
cpack(0x000000);
fill the polygon

Finally, all drawing states should be returned to reasonable values:

zbuffer(TRUE);
backbuffer(TRUE);
zdraw(FALSE);
wmpack(0xffffffff);

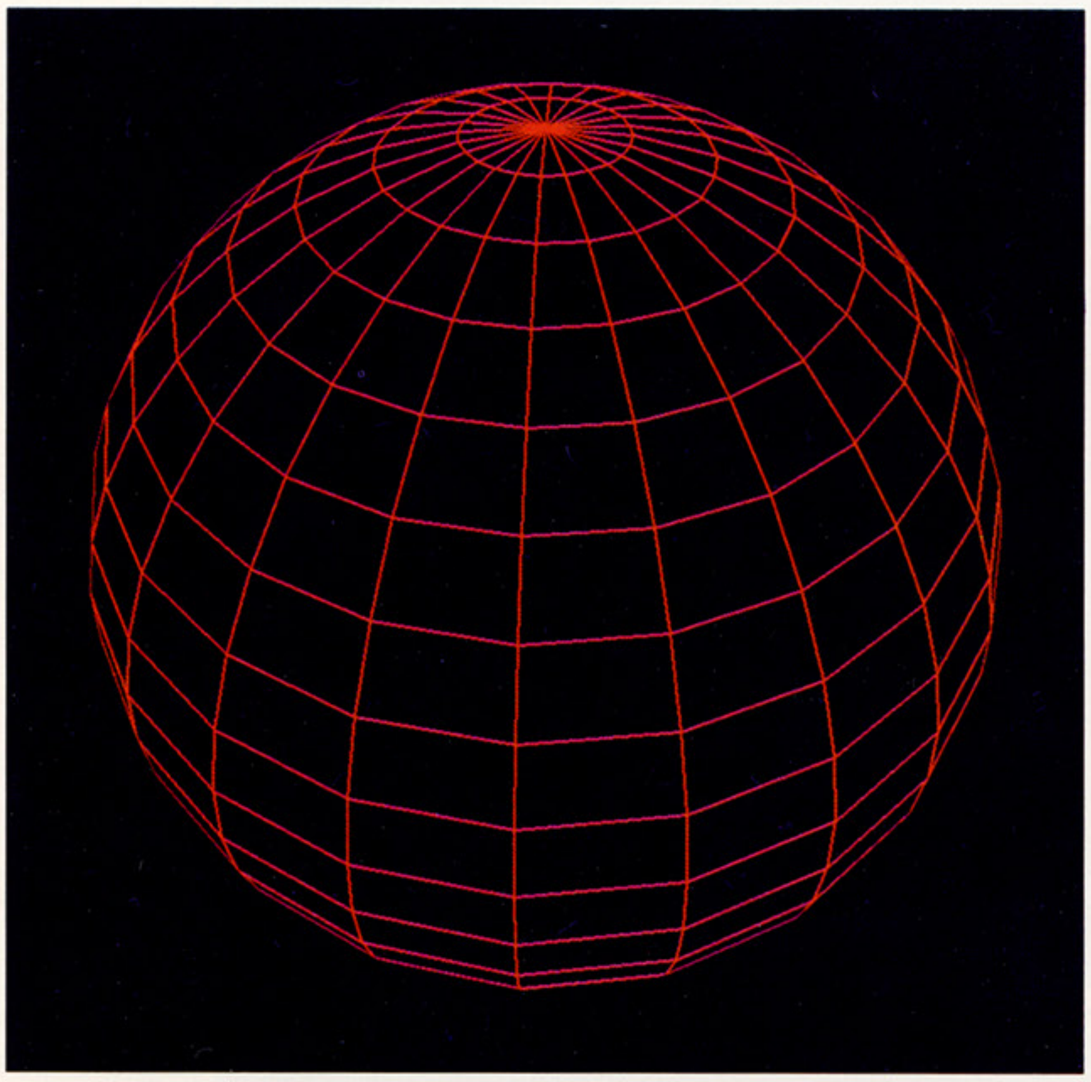


Figure 6: A hidden-line rendering of the solid, drawn with the hollow polygon technique, filling the polygons with background color.

When hollow polygons are drawn back-to-back, many of the mode changes in the first and last steps can be skipped, resulting in improved performance. Figure 4 contains a solid model of a sphere, with each facet outlined using standard Z-buffer techniques. Figure 5 shows what the same model looks like with the facets outlined using the hollow polygon technique.

Conclusion

We can draw the sphere (or any other solid) as a line drawing with hidden lines removed simply by filling the hollow polygons with background color, rather than with surface color. Figure 6 illustrates this effect.

Obviously, the GT/GTX Z-buffer gives us a powerful tool with applications beyond simple hidden-surface removal. In addition to the examples given here, programs have been written to perform constructive solid geometry, shadow casting, and more. What can *you* do with the Z-buffer? ■

Kurt Akeley is the Chief Engineer in Silicon Graphics' Advanced Systems Division. As one of SGI's founders, he has made numerous contributions to the evolution of the company's CPU and graphics subsystems technology.

Appendix B

When the projection matrix is defined by

perspective(*fovy*,*aspect*,*near*,*far*);

and the z viewport transformation is defined by

lsetdepth($near_{vp}$,far_{vp});

then z_{eye} and z_{screen} are related by the following equations:

$$z_{screen} = \left[\frac{far+near}{far-near} + \frac{2\ far\ near}{z_{eye}\ (far-near)}\right]\left[\frac{far_{vp}-near_{vp}}{2}\right] + \frac{far_{vp}+near_{vp}}{2}$$

$$z_{eye} = \frac{\dfrac{far\ near\ (far_{vp}-near_{vp})}{(far-near)}}{z_{screen} - \dfrac{(far+near)\ (far_{vp}-near_{vp})}{2(far-near)} - \dfrac{far_{vp}+near_{vp}}{2}}$$

Degenerating to Wireframe for Viewing

By Pramod Rustagi

During rotating and viewing operations, it is important to provide the user sufficient feedback (i.e., high frame-rates) to maintain an intuitive feel of the model orientation via the mouse or other input device. Regardless of the size of the model or performance of the underlying graphics hardware, the user must perceive that the response of the input device is in close relation to his movements. For a successful Spaceball™ implementation, high frame-rates are especially critical.

Furthermore, high end users of applications programs are frequently trying to build larger and more complex models which tax even the upper-end of available graphics performance. Today, the IRIS offers graphic performance of Z-buffered and lit polygons of 4.5K polygons for the 4D/20 and up to 100K for the GTX series.

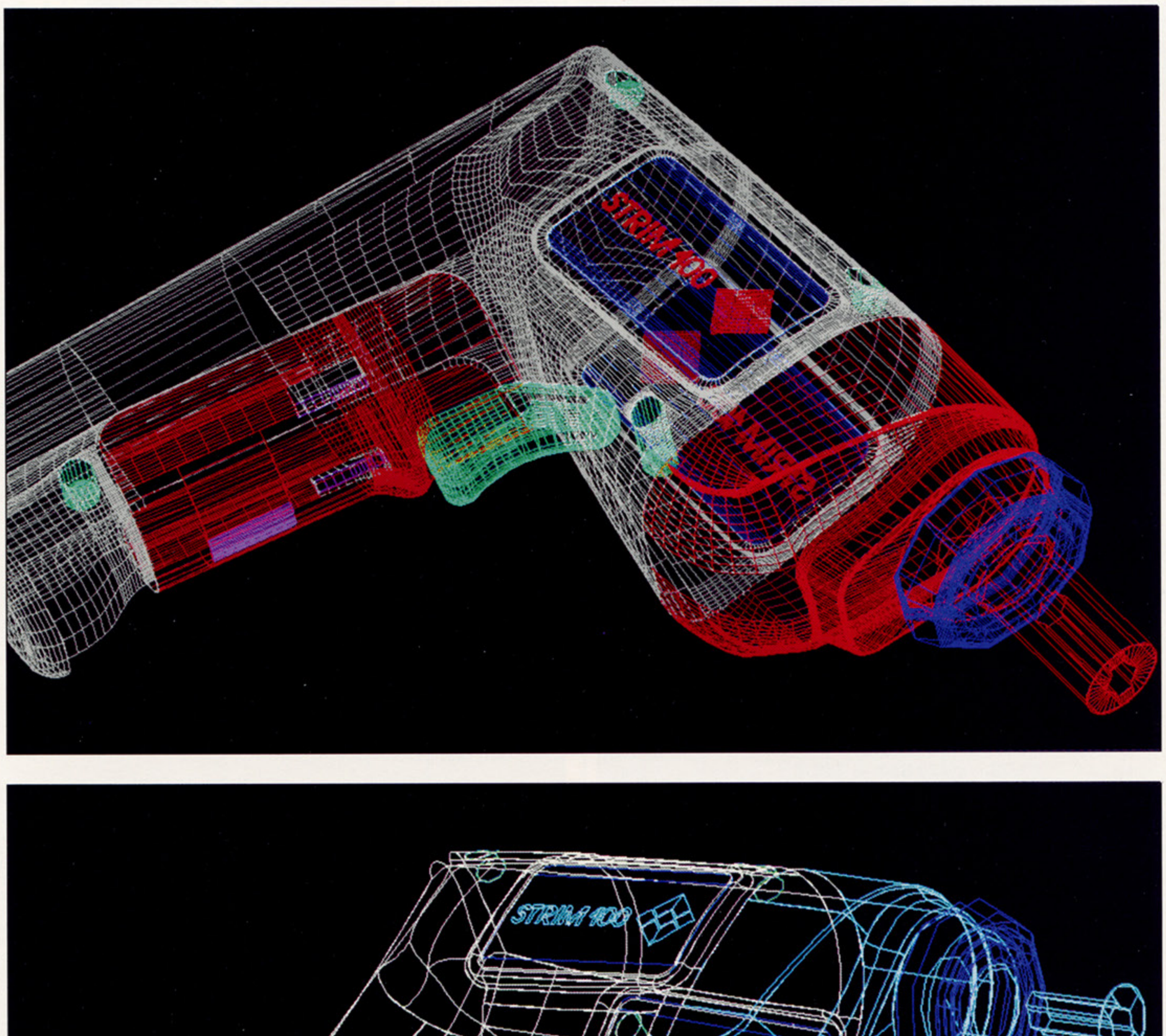
This issue is being approached by utilizing the viewing algorithm in an "adaptive" manner. Ideally, the technique determines the response that the underlying hardware is providing to the mouse on a specific model and then automatically chooses an appropriate representation or rendering of the model. For example rotating a "heavy" lit model of an automobile carburetor may take five seconds per frame but rendering the same model in wireframe may take only a fraction of a second per frame.

Short of being truly adaptive, however, we can provide a user selectable mode in which a "lightweight" model is used automatically when the user is performing viewing operations. The emphasis is on " automatic", such as with a mousebutton press, so that the user does not have to do extra work such as menu hunting. On a dial box, when the user is actively rotating a dial, the application program could draw in wireframe, and revert to heavy drawing when a small interval of time passes without dial events. Similarly, Spaceball events appearing in the input queue could trigger wireframe mode. To the user, it appears that the model "degenerates" into wireframe.

In the code sample on page 40, we show how to generate a wireframe image from the same data structures used to generate the heavy model. The normal lit model has a surface data-type that is traversed differently when in "degenerate" mode to render the wireframe quickly. This approach has the benefit of enabling the concept of wireframe degeneration without the necessity of creating alternate data-structures specifically for this purpose. ■

Pramod Rustagi manages the Applications Support Group at Silicon Graphics.

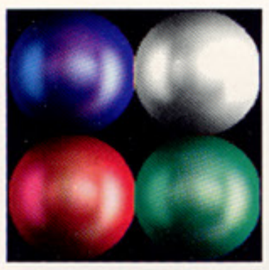
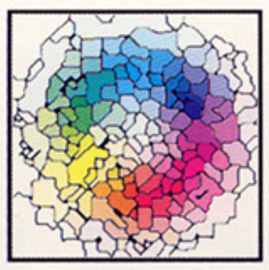
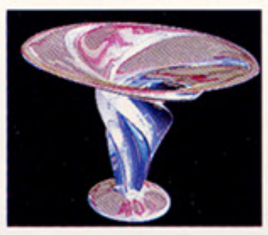
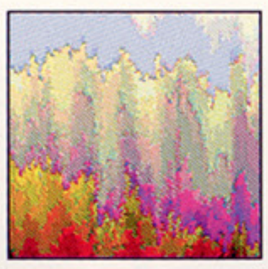
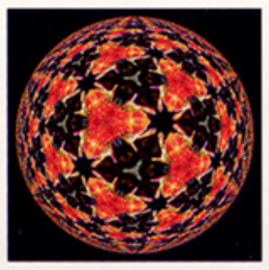
Right: a model of an electric drill is shown in various drawing modes; a "heavy" lit image composed of surface patches, a dense wireframe composed of all polygon edges of the surfaced lit model, and a lightweight wireframe composed only of edges on a surface patch. The geometric data of the model appears courtesy of Cisigraph Corporation.

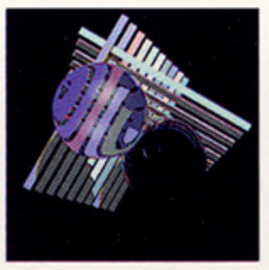
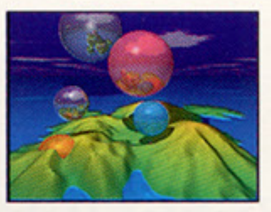

STRIM 400
STRIM 400
DOP MISTE

SOLUTIONS

```
qdevice(LEFTMOUSE);
  qdevice(MOUSEX);
  qdevice(MOUSEY);
  qdevice(TIMER0);
  noise(TIMER0, fraction_sec * HZ);
  .
while (FOREVER) { do {
  dev = qread (&val);
  switch(dev) {
  .
  case LEFTMOUSE:
    /* on mousedown, degenerate to
    lightweight */
    if(dgen_enabled) {dgen_it = TRUE; refresh
    = TRUE;}
    /* on mouseup, revert to heavy model */
    if(dgen_enabled) {dgen_it = FALSE;
    refresh = TRUE;}
    break;
  case MOUSEY:
  case MOUSEX:
    /* calculate delta mouse movement */
    break;
  case DIAL0:
    /* process dial event, dial delta,
    and provide for dial wrap here */
    time_out_flag = 0;
    if(dgen_enabled && !dgen_it) {
    dgen_it = TRUE; refresh = TRUE;
    }
    break;
case TIMER0:
    /* if 2 timer events occur without dial
    event, then
    revert to heavy model */
    if(++time_out_flag >= 2) {
    dgen_it = FALSE; refresh = TRUE;
    }
    break;
} /* switch */
} while (qtest()); /* do */
.
/* if delta mouse and/or dial motion, do
viewing-
  matrix ops and set refresh = TRUE here */
.
    if(refresh) {
      draw_scene(dgen_it)
      refresh = FALSE;
      }
   } /* while TRUE */
.
.
draw_scene(dgen_it)
{
  int nlen = 8; /* node length of 8 floats */
  /* assume linear list of op codes & data to
  define a graphics
    structure with surface data type of four-
    sided polygons,
    organized as ncol & nrow nodes composed of
    normal &
    vertex data in form of
    ((nx,ny,nz,na),(vx,vy,vz,va)).
    va & na allow for quad-word alignment. */
    .
    switch(op_code = *ptr++) {
    .
    case ALIGN: /* advance ptr to force quad
    alignment */
      ptr += *ptr;
      break;
    case SURFACE:
      switch(dgen_it) {
      case 0: /* draw filled surface */
        save_ptr = ptr;
        inc = nlen * ncol;
        /* insure contiguous addressing by
          drawing in row order */
        for(i=0;i<nrow-1;i++) {
          bgntmesh(); /* limit of 256 verticies
          */
          for(j=0;j<ncol;j++) {
            n3f(ptr); v3f(ptr+4);
            n3f(ptr+inc); v3f(ptr+inc+4);
             ptr += nlen;
          }
          endtmesh();
        }
        ptr = save_ptr + (nlen * nrow * ncol);
        break;
    case WIRE: /* draw surface in wireframe,
        use depthcuing, or lighted wires if
    desired */
      save_ptr = ptr;
      /* draw lines in row direction */
      for(i=0;i<nrow;i++) {
        bgnline();
        for(j=0;j<ncol;j++) {
          v3f(ptr+4); ptr =+ nlen;
        }
        endline();
      }
      .
      /* now draw in column direction */
      .
        ptr = save_ptr + (nlen * nrow * ncol);
        break;
      case NICE_OUTLINE:
        /* traverse only outer-edges of surface
      here */
        break;
      case RAW_OUTLINE:
        /* quickly draw lines only between outer
          four points of surface patch */
        break;
      } /* switch(dgen_it) */
    } /* switch(op-code) */
  }
```

COMMUNITY FORUM

New Software Tools for Four Color Separation

Software tools are available for IRIS 4D workstations to transform RGB images into color separations for four color printing. The 4D workstation used must have the ability to display images in RGB mode. Applied at SGI for the past three years, the software has been used to process over one-hundred images for internal publications and for this magazine.

To obtain *Creating Digital Color Separations on the IRIS* contact Monica Schulze, Silicon Graphics, 2011 N. Shoreline Blvd., Mountain View, CA 94039, (415) 962-3320, *monica@sgi.com*

Volume Visualization Workshop

Submission of papers and demonstrations of new techniques for visualizing data defined in volumes (three dimensianal arrays of data points) are being sought for a workshop to be held December 10-11, 1990 in San Diego, California. Data from any scientific or engineering application is welcome.

The Association of Computing Machinery (ACM) workshop is patterned after last May's Chapel Hill meeting and will be organized in cooperation with the San Diego Supercomputer Center, Molecular Graphics Society, and the National Science Foundation.

To receive a copy of the call for participation, contact: San Diego Workshop on Volume Visualization, San Diego Supercomputer Center, P.O. Box 85608, San Diego, CA 92138-5608; (619) 534-5000; email: *vvworkshop@sds.sdsc.edu*.

Joint Development Project

Silicon Graphics and Protocol Engines, Inc. have entered into a joint development project to implement the Xpress Transfer Protocol (XTP) for FDDI communications in VLSI.

XTP is a lightweight local area network (LAN) transport protocol that promises to deliver more highly effective throughput, real-time response, and lower CPU utilization than traditional transport protocols. By implementing the protocol in hardware, Protocol Engines is attempting to overcome the throughput limitations of software-based protocols. Initially, only software based implementations of XTP will be available, but a VLSI chip set is expected to be made available by late 1990.

Once completed, XTP will join the Silicon Graphics Visual Network product line. The product line allows for distributed data visualization by linking visualization software, high-performance networking, and multi-vendor workgroup connectivity. XTP has been formally proposed to ANSI Technical Committee X3S3.3 for consideration as a protocol standard.

For more information, contact: Jill Grossman, SGI, (415) 335-1516, or Larry Green, Protocol Engines, Inc. (703) 663-1571.

Silicon Graphics Brings Visual Processing to IBM

Silicon Graphics, now the leading manufacturer of Visual Processing systems, recently announced the licensing of its IRIS Graphics Library™ (IRIS GL) to International Business Machines. Silicon Graphics and IBM have worked closely to implement a compatible version of the IRIS GL, and will continue collaborating to ensure application compatibility in the future.

SGI will also supply IBM with special boards to facilitate the use of its GL-based, ultra-realistic graphics on IBM's new RISC System/6000 family. The 3D graphics processor will be offered as an option on all four of the new graphics workstations announced by IBM on February 15. The GL will come bundled with IBM's version of UNIX AIX.

"Silicon Graphics' strategy is to encourage the adoption of visual processing. We expect that working with IBM will accelerate the explosive growth of this marketplace," commented Edward McCracken, Silicon Graphics' president and CEO.

For more information, contact your sales representative or Todd Johnson, Silicon Graphics, 415/335-1166. ■

Submissions to Community Forum are welcome. Please send submissions to the attention of Gaye Graves, IRIS Universe, Silicon Graphics, Inc., 2011 N. Shoreline Boulevard, Mountain View, California 94039-7311, dougc@sig.com.

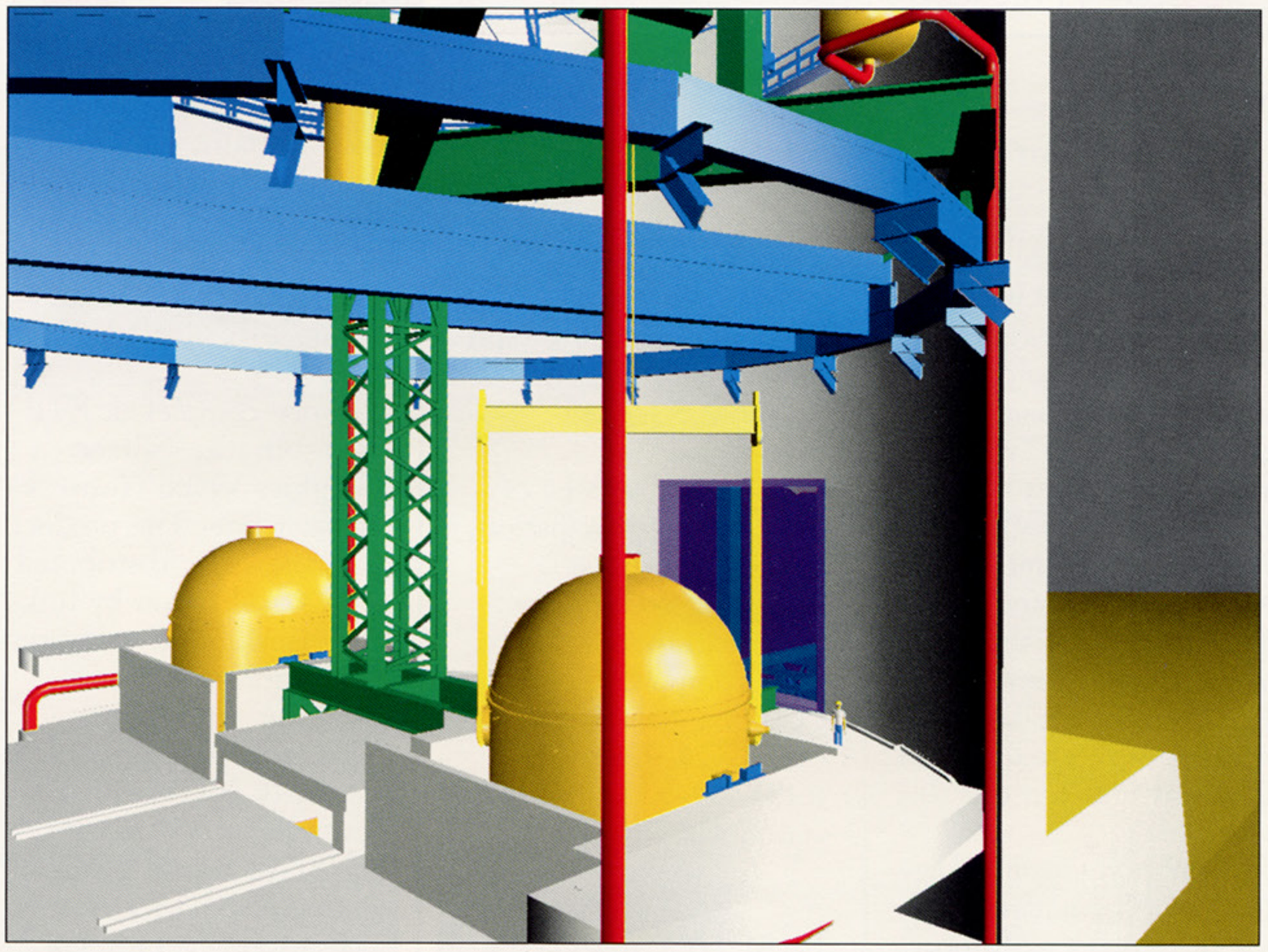
PRODUCT BRIEFING

Bechtel Releases WALKTHRU™ 3.0

With the release of the new version 3.0, Bechtel Software's WALKTHRU animation and visualization software now provides enhanced rendering and model annotation. WALKTHRU's new shading feature allows a user to determine up to eight local or infinite light sources — color, intensity and location are user-definable. Also included is a translucency feature for viewing internal structures or piping. WALKTHRU has been further improved with optional backface removal, and the ability to automatically annotate objects in a model with a text string entered by the user or automatically formed from components. Automatic placement of dimensions and dimension lines is yet another component new to version 3.0. All of these features are saved in an ASCII format for post-processing by the user's software.

WALKTHRU allows a user to view 3D computer models as though they already exist in the real world. By loading model files created in 3D onto WALKTHRU, a user can move about the model in real-time, controlling head and body movement. The operation of objects inside the model, all in fully shaded images, can also be simulated. WALKTHRU's intelligence detects any design interference in the model. All of these improvements increase the developer/client's interaction with the proposed model and his participation in the design process.

For more information, contact: A.B. Cleveland, Bechtel Software, Acton, MA, (508) 635-0580.



New CAD/CAM Applications Available

CADRA-III from Adra Systems, Inc. is a low-cost, professional drafting package which provides outstanding performance for design and production tasks. CADRA is similar to Cadam and can effectively transfer in both directions. CADRA is a complementary solution to many MCAE applications that require a low-cost, highly functional CAD package. For more information contact: Jim Clemens, (617) 891-8100.

Two new CAD/CAM products are now available from Gerber Systems Technology . Sabre-5000 offers excellent surface modeling capabilities along with manufacturing applications, and sheet metal simulation. Shoemaker, also from Gerber, is used for the 3D design, engineering, and manufacturing of footware. For more information contact: Meryl Charnow, (617) 891-8100.

New Mid-Range Workstation: 4D/85GT

Silicon Graphics has announced a new mid-range graphics workstation, the IRIS 4D/85GT. The 4D/85GT is fully binary compatible with all members of the SGI product family and offers a substantial price decrease over the same

level of performance on similar Silicon Graphics products. The system also provides the lowest cost entry point capable of upgrade into the GTX.

The 4D/85GT is rated at 13 MIPS, 1.5 MFLOP, 55,000 sustained polygons per second and includes a new expanded VME backplane. Standard disk drives and tape units are easily removable. The 4D/85GT is a powerful platform for the application developer due to the expanded 6 slot VME backplane, removable disk and tape units, and easy upgrade path.

For more information, contact: Jill Grossman, Silicon Graphics, (415) 335-1516.

Logicware's MPROLOG for the Personal IRIS

Logicware International's artificial intelligence language, MPROLOG, previously available on the IRIS 4D Power Series, is now available on the Personal IRIS under OS 3.1. MPROLOG is a Clocksin Mellish compatible implementation of Prolog suitable for building expert systems and large artificial intelligence applications. The more than one hundred built-in predicates can be extended through call-outs to C and Fortran.

Rapid program development is ensured through modular interfacing features and an interactive Program Development Support System with integrated editor, spy-point tracing, interactive execution, on-line help, and access to operating system commands. The MPROLOG System comes with a source library of utilities for mathematical functions, extended list handling, extended string handling, and Computer Aided Instruction software that teaches the Prolog language.

For more information, contact: Logicware International, Ontario, Canada, (416) 629-8801.

Accelr8 Product is Nominated for 1990 Target Award

Accelr8 Technology's LIBR8 has been nominated for a *Digital Review* 1990 Target Award. LIBR8 is one of Accelr8's line of VMS emulation and porting software products for UNIX systems. The LIBR8 Run-Time Library and System Services is an implementation of the functionality of the VMS Run-Time Library and System Service calls designed to run under various UNIX operating systems. Using LIBR8, one can quickly port large applications from a VMS environment to a UNIX environment.

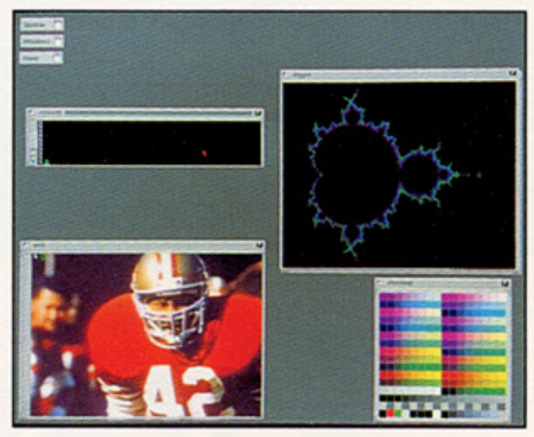
The LIBR8 package provides functions on UNIX computers that previously were available only on Digital Equipment Corporation VAX computers running VMS. These software libraries create a migration path for users who faced significant program redesign barriers when moving VMS to UNIX. Customers who will benefit from the LIBR8 libraries are those with features such as MAILBOXES, EVENT FLAGS, and QIO routines in their programs.

For more information, contact: Robert W. Hickler, Accelr8 Technology, Denver, CO, (303) 863-8088

A New Addition to RGB Spectrum's Line of Video Windowing Systems

RGB Spectrum's new Model 1000, like the model 2000, integrates real time video with text and graphics on a workstation monitor . The price, however, is significantly lower than that of the Model 2000. Real-time video (NTSC or PAL) is displayed as a window on the workstation screen. The video can now be positioned, scaled, clipped, and overlaid with computer graphics.

All functions are accessible via an RS-232 port that allows control by the workstation. The video window may be manipulated under the windowing system. The display output is directed to the computer's own high resolution RGB monitor or to a high scan rate video projector.



The RGB/View 1000 is both frame buffer and host independent, and does not affect the workstation's CPU or frame buffer performance. The Model 1000 may also be connected to any camera, tape recorder, live television, interactive video disc or video teleconferencing system. For more information, contact: Carol Fogel, RGB Spectrum, Berkeley, CA, (415) 848-0180.

COMING ATTRACTIONS

Coming in the Next Issue of *IRIS Universe:*

Computer Graphics at Lucasfilm
The magicians at Industrial Light and Magic conjure up custom programs to achieve film directors' visions.

Sculpture from Computers
With photo-sensitive liquid resin and a computer-directed laser, a Southern California man is pushing modern art into the future.

Black Box Simulations
The Bureau of Air Safety Investigation in Canberra, Australia, is making innovative use of 3D simulations to study, and prevent, aircraft accidents.

Putting a Human Face on Computers
The trauma a patient may experience during recovery from reconstructive surgery can be minimized by using data from CAT-scans to develop 3D computer visualizations.

Coming in Future Issues:

A Comparison of Windowing Environments

Geo Sciences: The Present and the Future

New Possibilities in Character Animation

A Random, Real-time, Walk Through Manhattan

Inside a Super Nova

Advanced Beta-Spline Techniques

Virtual Reality: A Day in the Life of a Parallel Universe

AND MUCH MORE!

SILICON GRAPHICS, INC.
EDUCATION CENTER
COURSE CALENDAR
(Through June 1990)

COURSE	*LOCATION**	
4D SERIES COURSES	WEC	EEC
Graphic Programming† *4.5 days*	Mar 19, 1990 May 14, 1990 June 25, 1990	Apr 2, 1990 April 30, 1990 June 4, 1990
Advanced Graphics *3.5 days*	Mar 26, 1990 Apr 23, 1990	May 7, 1990
Parallel Programming *4.5 days*	Mar 12, 1990 May 21, 1990	not available
System Accelerator *4.5 days*	June 4, 1990 Apr 2, 1990	Mar 5, 1990 Apr 9, 1990 June 11, 1990
System Administration *4.5 days*	Apr 9, 1990 Jun 11, 1990	Mar 12, 1990 June 18, 1990
Network Administration *4.5 days*	Apr 16, 1990	Mar 19, 1990
System Maintenance *10.0 days*	June 11, 1990	Apr 16, 1990
Personal IRIS Maintenance *3.5 days*	Apr 9, 1990	not available
Multiprocessor Maintenance *4.0 days*	Mar 26, 1990 June 25, 1990	not available

KEY: WEC—Western Education Center, Mountain View, CA.
EEC—Eastern Education Center, SGI Federal, Bethesda, MD.

*The SGI Education Center reserves the right to cancel classes due to insufficient enrollment.

To register or obtain more information, call 800/356-9492.

†Effective July 1, 1990 Tuition for Graphic Programing will increase.

Show Your True Colors

Introducing the GBA-Scanin Subsystem, a high resolution, true color scanner application for use with Silicon Graphics' 4D Family of computers.

GBA-Scanin gives you high-end scanning power with the SHARP JX-450 and JX-600. The 11"x17" scanning bed even works with transparencies — 8.3"x11.7" on the JX450, and 11"x16.5" on the JX600 with scanning resolution up to 600 dpi .

When used with SHARP's new, compact JX-300, GBA-Scanin will give you clear, crisp images from originals up to 8.5"x11".

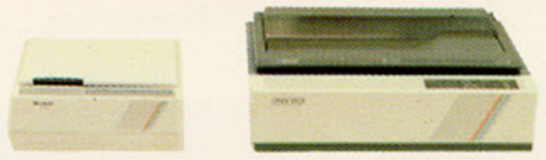
GBA-Scanin's flexibility lets you scan either an entire image, or a specific area from an image. Then save in your choice of file formats — SGI, WAVEFRONT, ALIAS, TIFF, PICT2, TGA, or ARTISAN.

You can get the entire Subsystem — scanner, GBA-Scanin, GPIB board, and a 4M cable — at a price lower than you'd ever expect.

Don't you think it's time to show *your* true colors?

SCANNING SUBSYSTEM

GBA-Scanin



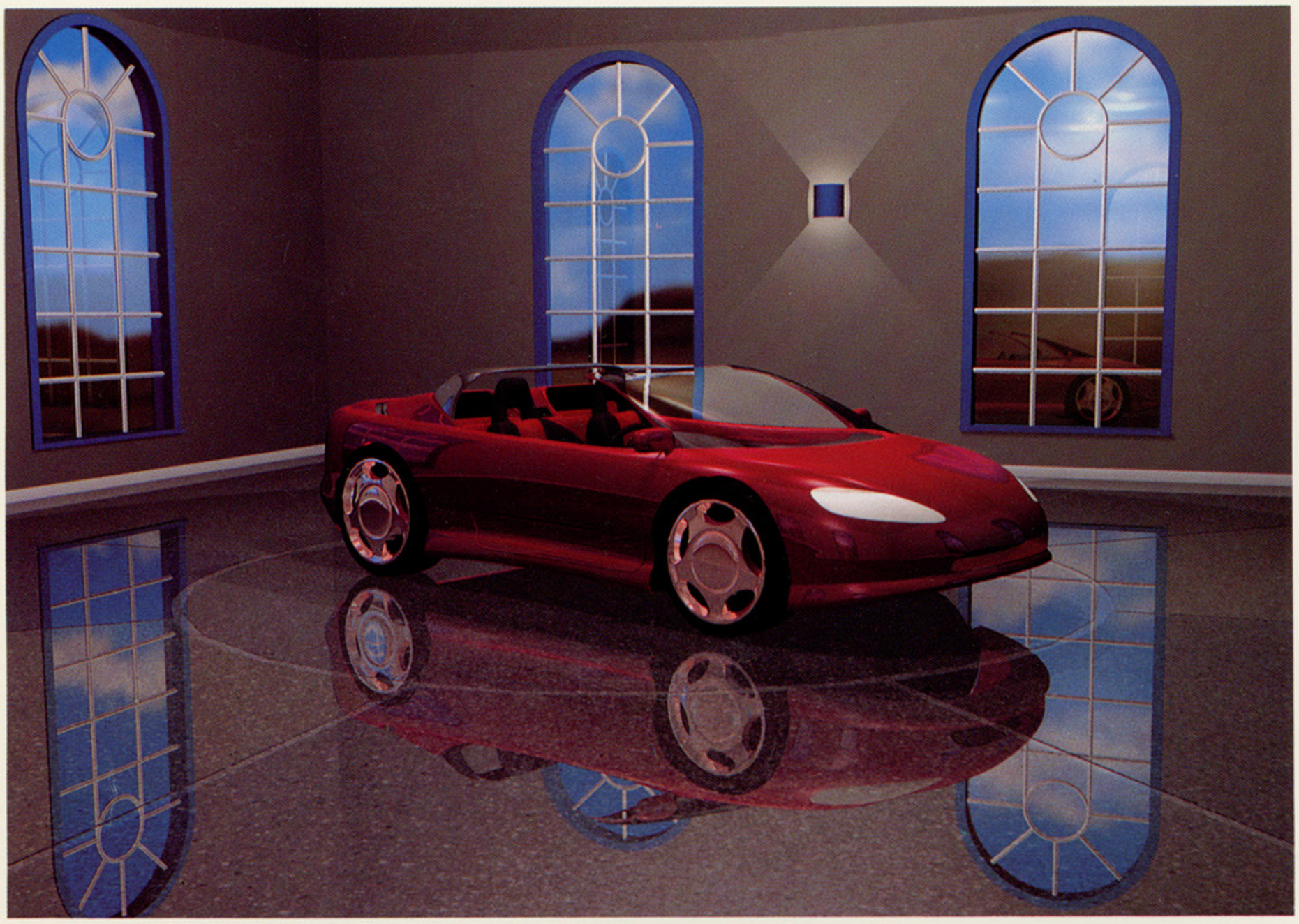
SHARP JX-300 SHARP JX-450/600

GBA

228 HAWTHORNE AVENUE
LOS ALTOS, CALIFORNIA 94022
415-948-4711
FAX 415-949-5005 or 408-370-2121

ALIAS™. Also known as "designing the future faster."™

From New York to Paris to Tokyo, hundreds of leading designers have one thing in common: they all depend on ALIAS to help design the future faster. Designers at companies like Sony, Honda, Oneida, Kraft, Timex, Johnson and Johnson, Reebok, Goodyear and Apple Computer - to name just a few.



With ALIAS you design more, design faster, and communicate your vision better. Your concept is always alive as you refine and change your "3D blueprint", quickly and precisely changing shapes, positions, material properties and colors. And most important, ALIAS is easy to learn and easy to use.

With ALIAS you also communicate good design at a business level. Presentations to management and clients come alive with video, color printouts or 3D laser prototypes. And you always design for manufacture with ALIAS, as you create an accurate 3D database for downstream analysis and manufacturing.

Years of experience at some of the world's most respected design studios has made ALIAS the #1 software solution for industrial design, worldwide. If you would like to learn more about how you can benefit from ALIAS, please call or write for a free informative videotape. And get prepared to design the future faster.

Alias

110 Richmond Street East,
Toronto, Canada M5C 1P1
Fax (416) 362-0630
Tel (416) 362-9181

The above image was computer generated with ALIAS™ software.